ENHANCING DYNAMICS COURSES WITH MODEL ELICITING ACTIVITIES

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Mechanical Engineering

by

Lawrence Fong

November 2009

COMMITTEE MEMBERSHIP

TITLE:	Enhancing Dynamics Courses with Model Eliciting Activities
AUTHOR:	Lawrence Fong
DATE SUBMITTED:	11/02/09
COMMITTEE CHAIR:	Brian Self
COMMITTEE MEMBER:	Andrew Kean
COMMITTEE MEMBER:	James Widmann

ABSTRACT

ENHANCING DYNAMICS COURSES WITH MODEL ELICITING ACTIVITIES

Lawrence Fong

Model eliciting activities are assignments which require students to develop models to describe realistic situations. Every MEA follows six principles: model-construction, reality, self-assessment, model documentation, generalizability, and effective prototype. The six principles provide a solid guideline in which instructors can develop more MEAs, which can then be shared and used among several participating universities. Under NSF CCLI Grant #0717595, Cal Poly is currently developing Model Eliciting Activities for the subject of Mechanical Engineering.

This report documents the undertakings to implement and enhance two Model Eliciting Activities (MEAs) into the Cal Poly curriculum. Specifically, the development of the Vehicle Accident Reconstruction (VAR) MEA and the Catapult MEA will be covered in detail.

The VAR MEA was a project assigned in ME212 "Engineering Dynamics," which required students to apply momentum principles to a two-vehicle collision. Because of the heavy development time experienced by the MEA research team, a MatLab program which accepted user inputs via a graphical user interface (GUI) was developed. This GUI solved for initial velocities during two-vehicle collisions by applying appropriate momentum and work-energy principles. With this program, instructors can more easily develop crash scenarios, as well as check student work.

The Catapult MEA was also a project assigned to ME212 students. It required them to analyze the launch trajectory of an actual scaled catapult using angular motion and work-energy principles. This scaled-catapult was instrumented with one ADXL278 dual-axis accelerometer and four CEA-06-240UZ-120 strain gages. This instrumentation allowed for the experimental data acquisition of the catapult angular velocity, acceleration, and strains. By postprocessing this experimental data using a MatLab program, the experimental results can then be compared to theoretical results.

The overall goal for the VAR MEA GUI programming was to reduce instructor workload in order to promote usage the MEA through a broader range of universities. The goal of the Catapult instrumentation was to provide students with actual experimental data, which could then be used to confirm their theoretical model. The system was set up so that they could easily record their own experimental data for each catapult launch.

ACKNOWLEDGMENTS

I would like to thank all the members of my thesis committee for their guidance and support during this past year. I would also like to acknowledge NSF CCLI Grant #0717595 which made this research possible.

Table of Contents

List of Figures

List of Tables

Nomenclature

α	angular acceleration
a_n	normal acceleration
a_t	tangential acceleration
ε	strain
E	elastic modulus
μF	capacitance (micro-Farad)
F	force
H	angular momentum
I	second moment of inertia
m	mass
M	moment
σ	stress
r	radius
S_g	gage factor
t	time
v	velocity
V	voltage
ω	angular velocity

Introduction

The problem solving aspect of engineering classes has always been an emphasis at Cal Poly. With Cal Poly's "Learn by Doing" philosophy, students are expected to possess fundamental engineering knowledge and design intuition. However, the current coursework assigned to students often omits exercises that nurture real-world analysis. Most textbooks require students to have only a superficial understanding of equations and symbols – without a deep conceptual understanding. By implementing Model Eliciting Activities (MEA) into the Dynamics curriculum, we hoped to enhance student learning and overall student performance.

A Model Eliciting Activity aims to build solid engineering fundamentals for students by requiring them to analyze open-ended scenarios and apply appropriate analysis. Every MEA follows six principles: model construction, reality, generalizablility, self-assessment, model-documentation, and effective prototype. By following these principles during problem development, we ensure that students are presented with a realistic client-driven problem which solidifies engineering principles and is applicable towards other situations. Cal Poly is currently responsible for the development of Mechanical Engineering MEAs in NSF CCLI Grant #0717595: *Collaborative Research: Improving Engineering Students' Learning Strategies through Models and Modeling*. The overall goals of this grant include expanding MEA usage into more universities and disciplines as well as analyzing the effect of MEAs on student learning. At Cal Poly, we have been working primarily on developing MEAs for use in sophomore and junior level Mechanical Engineering courses. These activities are currently being implemented into some sections of Engineering Dynamics (ME212), Thermodynamics I (ME302), and Thermal System Design (ME440).

In this paper, I will focus primarily on the Dynamics MEA development within the Mechanical Engineering Department at Cal Poly. In particular, the vehicle accident reconstruction (VAR) and catapult MEAs will be covered in extensive detail. The development of these MEAs constituted a large portion of my work in the MEA research team. This work includes the instrumentation and interface of the catapult, as well as programming of a MatLab GUI (graphical user interface) for the vehicle accident reconstruction MEA.

We propose that adding MEAs to the Cal Poly curriculum does in fact boost student understanding of engineering fundamentals. The MEAs have been evaluated every quarter through student surveys and exam performance. Exam scores indicate a possible increase in student performance in the conceptual areas reinforced with MEAs. Surveys indicate that despite the increased workload, students did in fact enjoy these projects. Because of this positive response, we have been encouraged to further develop the MEAs that will be discussed in this paper.

Model Eliciting Activities and Cal Poly

What is an MEA?

Model Eliciting Activities, which were first started in the mathematics community, are team-based activities which require students to analyze real-world, open-ended problems. Figure 1 highlights the difference between traditional word problems and Model Eliciting Activities. Traditionally, students are asked to solve problems mathematically and apply their solution to the real world. In contrast, MEAs require students to derive mathematical models from realistic situations.

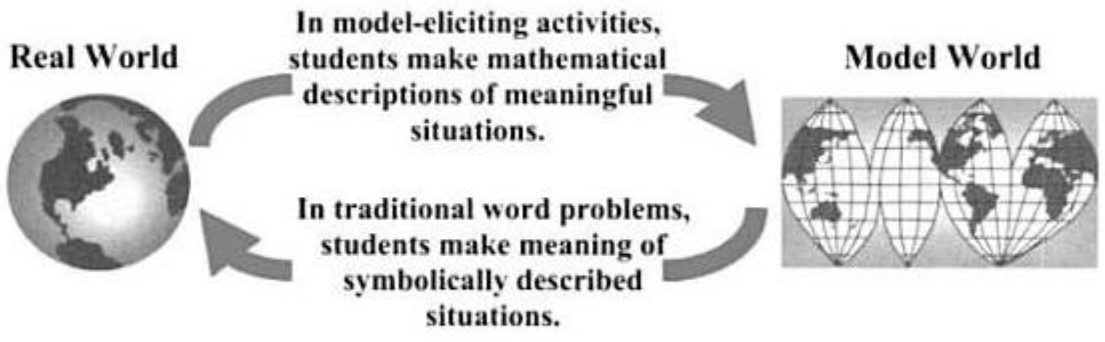

Real World
In model-eliciting activities, students make mathematical descriptions of meaningful situations.
Model World
In traditional word problems, students make meaning of symbolically described situations.

Figure 1. Difference between traditional word problems and Model Eliciting Activities. (Lesh, Beyond Constructivism: Models and Modeling Perspectives on Mathematics 2004, 4)

In a report from the Carnegie Foundation, "Reinventing Undergraduate Education: A Blueprint for American's Research Universities," an academic bill of rights for students is presented. Some of these rights include: "(1) Providing opportunities to learn throughout inquiry rather than simple transmission of knowledge, (2) Training in the skills necessary for oral and written communication, and (3) Preparing students carefully and comprehensively for whatever may lie beyond graduation" (Boyer Commission on Education Undergraduates in the Research University 1998, 12). The goals of MEAs are closely aligned with these rights, and are reflected in its six principles.

The following are those six principles that every MEA should follow, which provides an instructor's guideline for problem development (Self 2007). Each one of these principles serves to promote a more applicable type of learning for students. These six principles are summarized below:

1. The <u>Model-Construction Principle</u> requires students to develop a mathematical system as a deliverable to an indicated client.
2. The <u>Reality Principle</u> requires the activity to be set in a realistic engineering setting, and allows students to connect their real-world experience to the problem. Students should be allowed and encouraged to make realistic assumptions based on their existing knowledge.
3. The <u>Self-Assessment Principle</u> allows students to evaluate their own work and revise their models accordingly. Students should be encouraged to test their models and improve them for their client. They should also be able to assess when their work is complete.
4. The <u>Model Documentation Principle</u> requires students to carefully detail their process in developing the model. Typically this includes a memo to their client describing a walkthrough of their analysis. This allows both instructors and students alike to see a logical progression of the model, and to see the thought process behind it. From this, instructors can more easily identify any areas of difficulty students have.
5. The <u>Generalizability Principle</u> requires students to develop models that have a value outside of a specific scenario. These models should be easily modified and applicable to similar scenarios outside of the ones that were assigned.

6. The Effective Prototype Principle requires that the developed models have an intellectual significance and impact on the future professional lives of students. The models should provide for a useful mental foundation to interpret similar situations in the future.

MEAs go beyond the commonly requested numerical answers that are so commonly asked from students. For most problems or exercises that are presented in textbooks, the student is merely required to reproduce a brief answer to a question that was formulated by others (Lesh, Handbook of Research Design in Mathematics and Science Education 2000, 594). This results in only a superficial understanding of the material – the student disregards his process and focuses instead on if his answer was correct.

Difference between MEAs and Traditional Assignments

Instead of basing learning on the "correctness" of the final answer, MEAs require students to focus on the *method* that they use to arrive at their solution. The description, explanations, and constructions are not simply processes that students go through in order to produce final answer – they are the most important aspect of their analysis (Lesh, Beyond Constructivism: Models and Modeling Perspectives on Mathematics 2004). Since these activities are also team based, students are also exposed to working in small groups. In this team environment, students are expected to eloquently share their ideas with other members, and work cohesively to produce a working model.

As will be discussed in further detail later in the paper, the VAR and catapult MEAs were evaluated not primarily on correctness of a student team's final answer, but on the thought process that they carefully documented through the project. By requiring careful model documentation, we were able to more easily identify student misconceptions – allowing instructors to allocate more time to areas of student difficulty.

History of MEAs

The concept of Model Eliciting Activities is not a new one – problem based learning (PBL) has existed since the 1960s, and has garnered much support from educators. A PBL is defined as “an instructional learner-centered approach that empowers learners to conduct research, integrate theory and practice, and apply knowledge and skills to develop a viable solution to a defined problem” (Savery 2006). Problem based learning shares a great number of similarities with MEAs. These similarities include realistic problems, open-ended tasks, higher order thinking, self-directed learning, self-assessment, group work, and structure of the problems (Chamberlin 2008).

Because of these similarities, several parallels can be drawn from PBL to MEAs. Although the large variations in the practicing of PBLs make the analysis of its effectiveness difficult, one of the most widely accepted findings is that PBL promotes positive student attitudes (Prince 2004). In our own experience, we have found that student attitudes and performance have been improved by implementing MEAs at Cal Poly. In addition to this positive benefit on students, MEAs set forth a solid structural framework, which is used as criteria for instructors to develop new MEAs. Although not much data on MEA effectiveness is currently available, this framework provides a unified guideline so that MEAs across universities can be compared. In this manner, correlations between student performance and MEA implementation can be more easily drawn.

Research Team

The Cal Poly MEA research team is part of a four-year effort by a team of researchers from seven universities. These researchers utilize previous mathematics MEA development as a foundation for undergraduate STEM curriculum and assessment for engineering (Self 2007). At Cal Poly, the goal is to develop new Mechanical Engineering MEAs for implementation into either laboratory activities or in-class projects.

The MEA research team at Cal Poly currently consists of a combination of professors and students. The past and current participants are listed in the following table. Every week the MEA team met to discuss future and present MEAs. This entailed discussing the implementation of current projects, student difficulties, and potential future projects.

Table 1. Roster of MEA team.

		Academic Year	
Name	Position	2007-2008	2008-2009
Brian Self	Professor	Fall, Winter, Spring	Fall, Winter, Spring
Andrew Kean	Professor	Fall, Winter, Spring	Fall, Winter, Spring
Jim Widmann	Professor		Fall, Winter
Lawrence Fong	Graduate Student		Fall, Winter, Spring, Summer
Teresa Ogletree	Undergraduate Student	Summer	Fall, Winter
Lora Powers	Undergraduate Student		Fall, Winter
Frank Schreiber	Undergraduate Student	Spring	Fall, Winter, Spring
Annamarie Usher	Undergraduate Student		Spring
Rosalie Mangione	Undergraduate Student		Spring

Cal Poly and MEAs

NSF CCLI Grant #0717595 lists Cal Poly as the prime on developing Model Eliciting Activities for Mechanical Engineering. This entails developing MEAs in common disciplinary topics such as fluids, thermodynamics, energy conversion, heat and mass transfer, mechanics, and structural analysis, in addition to machine design (Self 2007). Of these possible topics, we chose to start with ME212, because it has a very broad engineering student population and is also a very problematic class in terms of fail rate.

It is also one of the most demanded classes – up to 9 sections of over 30 students each are taught each quarter. Nearly all engineering majors are required to take ME212 during their career at Cal Poly, resulting in a very diverse group of students within each class. Future MEA developments are also targeted at these sophomore-level courses because they have a broad audience and can be easily distributed to other engineering universities for use in their curriculum. At Cal Poly, the MEAs we have generated have followed this basic structure:

Instructor Provides:

- Some background information is provided using a current news excerpt or headline. This makes students understand the significance of their efforts and allows them to put their analysis into a real-world context.
- A client requests the students to develop a procedure for solving a particular engineering issue. This is typically set in a professional tone – using a company memo.

Student Provides:

- Detailed methodology to solve the engineering problem.
- Supporting calculations to demonstrate the application of their engineering process.
- Summary in memo format.

MEAs in Cal Poly Dynamics Courses

Cal Poly lists ME212, “Engineering Dynamics”, as a course which focuses on the concepts of velocity, acceleration, relative motion, work, energy, impulse, and momentum. As mentioned previously, MEAs were first implemented in this class because of its high failure rate and broad student population. In Cal Poly’s quarter system of 10 weeks, students often struggle to fully understand each of these concepts – resulting in poor performance. Some professors indicate failure rates of approximately 15-30%.

Instructors from other universities also observed this problem and attempted to combat it in different ways. For example, in Worchester Polytechnic Institute in Massachusetts, instructors integrated the use of LEGO® kits into an introductory Dynamics course. Students were required to develop models to describe the kinematics and kinetics of a linkage. Since a major difficulty of learning Dynamics is caused by the lack of a physical model, this hands-on approach was seen as a great tool for learning (Jolley 2003).

Another example is Grand Valley State University’s catapult-design contest. Here, students were required to design and build a catapult to clear a vertical height and hit a target at a specified distance (Reffeor 2002). This required selection of materials, springs, and associated calculations. Shown in Figure 2 is a student-built catapult from the competition.

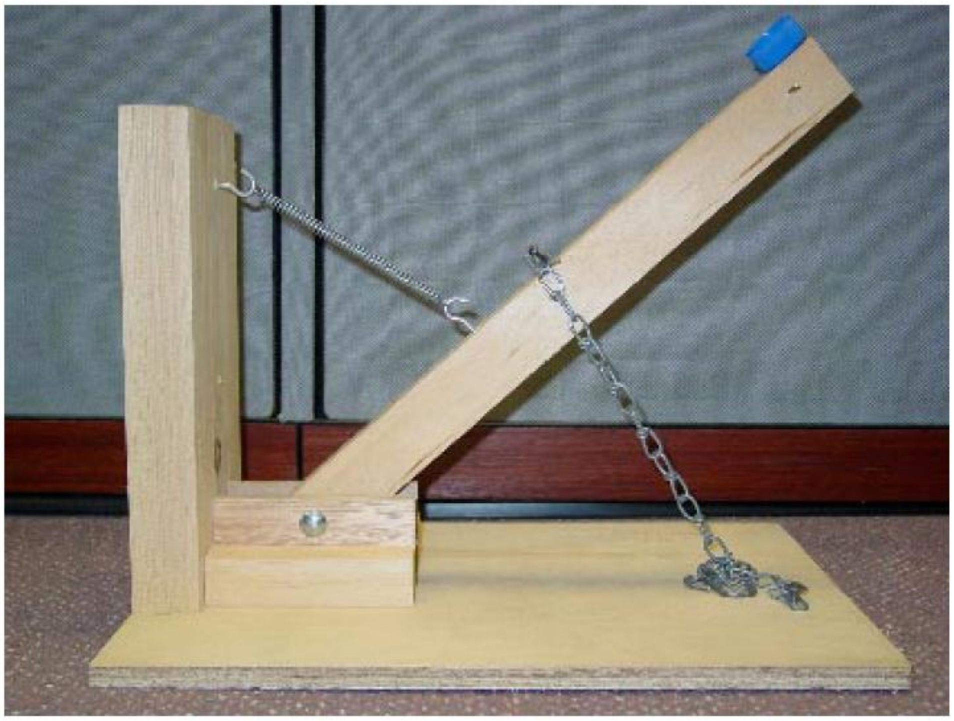


Figure 2. Catapult built by Grand Valley State University students.

Students at Grand Valley were critiqued on the correlation between their theoretical predictions and the actual results. As will be discussed in the Catapult section, the Catapult MEA at Cal Poly was similar to this project. However instead of requiring students to actually build the catapult, we instead focused on the development of the theoretical model and the comparison to the physical results. We have found that MEAs can be very time intensive, so simpler MEAs that convey the same idea can be beneficial to students and teachers alike. In this manner, more subject matter can be taught with a wider variety of projects.

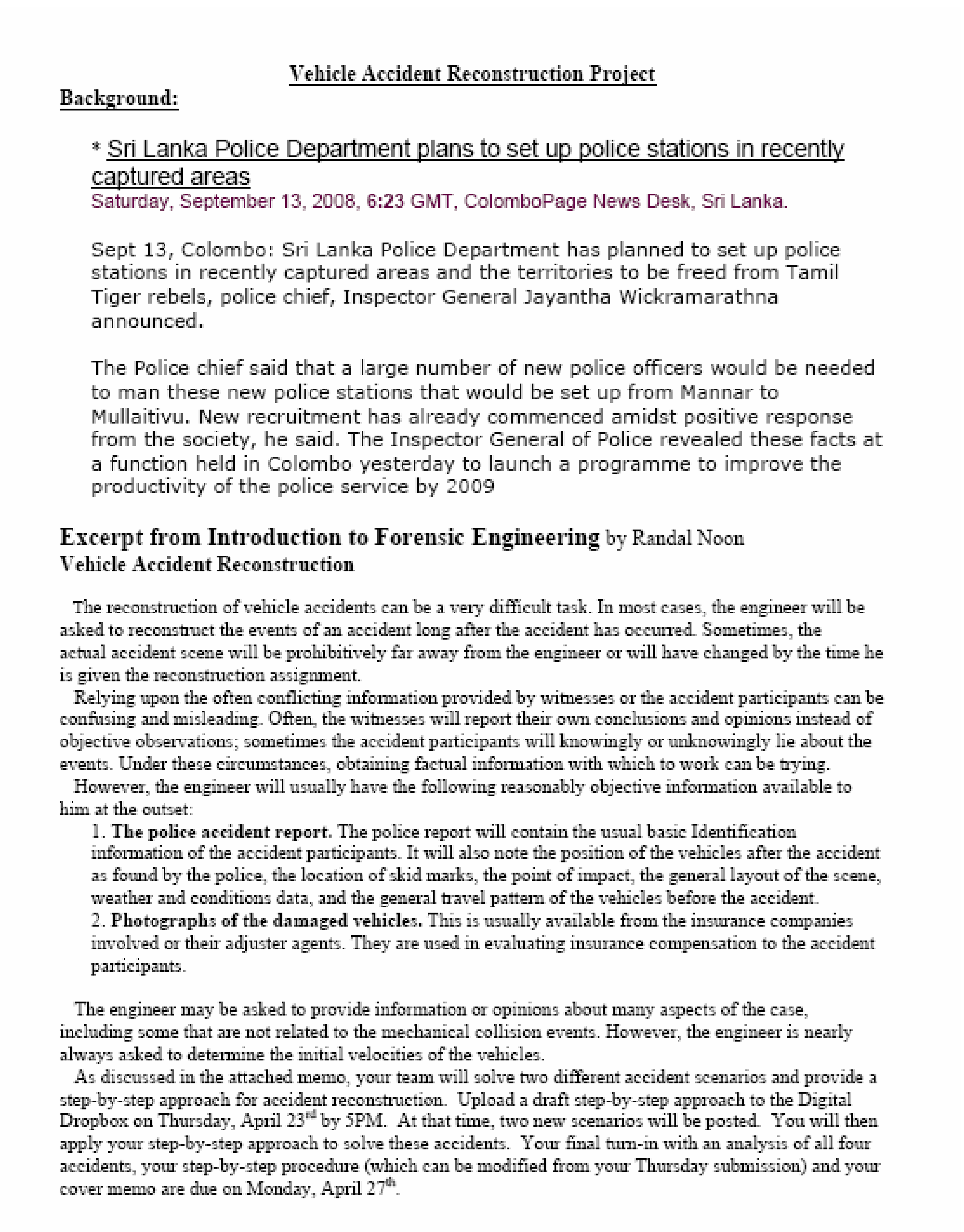
Our overall goal was to motivate students by providing a realistic project with the VAR MEA and a very hands-on project with the Catapult MEA. By doing so, we hoped to see increased student performance and willingness to learn.

Vehicle Accident Reconstruction

Application of momentum principles is one of the fundamental concepts introduced in early physics courses, and solidified in ME212. Because of students' previous exposure to the topic of momentum, and its direct applicability toward real-world scenarios, we developed an MEA to further solidify this concept with students.

The vehicle accident reconstruction project (VAR) was the first MEA that was developed by the research team at Cal Poly. During the fall quarter of 2008, with the newly assembled team, the VAR MEA was refined and assigned to the first dynamics class. The client of this MEA was a Sri Lanka police station which was developing an investigation protocol to determine fault in vehicular collisions. We chose this particular context in the hopes that it would capture the interest of students by including engineering analysis with a meaningful social impact.

Listed in the following figures are the background information and memorandum handouts given to students. The background information serves to provide students with preliminary information, pertinence to current events, and importance of their analysis. The memorandum presents a client-driven problem in a professional tone – setting up students to appropriately develop their model.



Vehicle Accident Reconstruction Project

Background:

* Sri Lanka Police Department plans to set up police stations in recently captured areas

Saturday, September 13, 2008, 6:23 GMT, ColomboPage News Desk, Sri Lanka.

Sept 13, Colombo: Sri Lanka Police Department has planned to set up police stations in recently captured areas and the territories to be freed from Tamil Tiger rebels, police chief, Inspector General Jayantha Wickramarathna announced.

The Police chief said that a large number of new police officers would be needed to man these new police stations that would be set up from Mannar to Mullaitivu. New recruitment has already commenced amidst positive response from the society, he said. The Inspector General of Police revealed these facts at a function held in Colombo yesterday to launch a programme to improve the productivity of the police service by 2009

Excerpt from Introduction to Forensic Engineering by Randal Noon

Vehicle Accident Reconstruction

The reconstruction of vehicle accidents can be a very difficult task. In most cases, the engineer will be asked to reconstruct the events of an accident long after the accident has occurred. Sometimes, the actual accident scene will be prohibitively far away from the engineer or will have changed by the time he is given the reconstruction assignment.

Relying upon the often conflicting information provided by witnesses or the accident participants can be confusing and misleading. Often, the witnesses will report their own conclusions and opinions instead of objective observations; sometimes the accident participants will knowingly or unknowingly lie about the events. Under these circumstances, obtaining factual information with which to work can be trying.

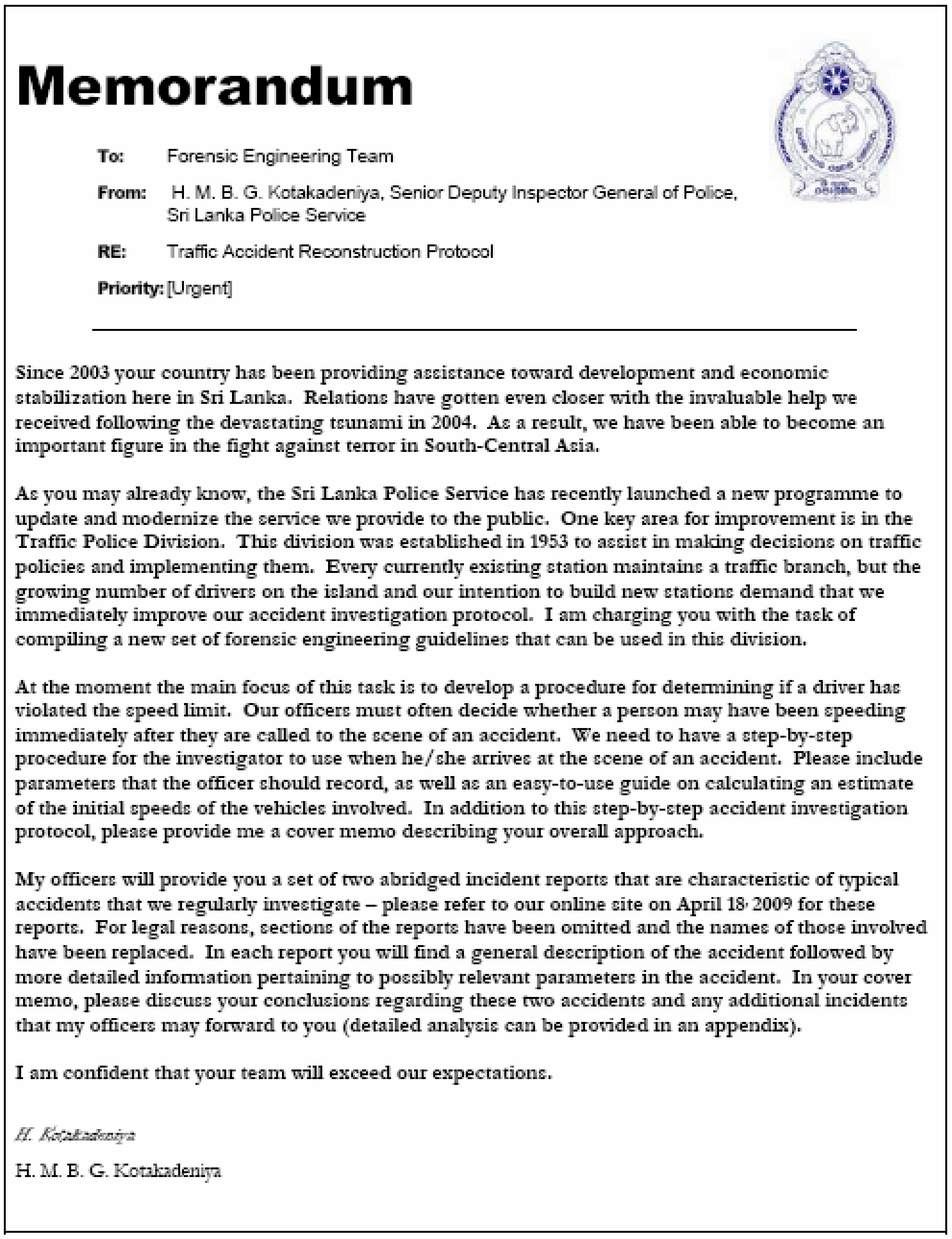
However, the engineer will usually have the following reasonably objective information available to him at the outset:

1. **The police accident report.** The police report will contain the usual basic Identification information of the accident participants. It will also note the position of the vehicles after the accident as found by the police, the location of skid marks, the point of impact, the general layout of the scene, weather and conditions data, and the general travel pattern of the vehicles before the accident.
2. **Photographs of the damaged vehicles.** This is usually available from the insurance companies involved or their adjuster agents. They are used in evaluating insurance compensation to the accident participants.

The engineer may be asked to provide information or opinions about many aspects of the case, including some that are not related to the mechanical collision events. However, the engineer is nearly always asked to determine the initial velocities of the vehicles.

As discussed in the attached memo, your team will solve two different accident scenarios and provide a step-by-step approach for accident reconstruction. Upload a draft step-by-step approach to the Digital Dropbox on Thursday, April 23rd by 5PM. At that time, two new scenarios will be posted. You will then apply your step-by-step approach to solve these accidents. Your final turn-in with an analysis of all four accidents, your step-by-step procedure (which can be modified from your Thursday submission) and your cover memo are due on Monday, April 27th.

Figure 3. Background information provided for VAR MEA.



Memorandum

To: Forensic Engineering Team

From: H. M. B. G. Kotakadeniya, Senior Deputy Inspector General of Police, Sri Lanka Police Service

RE: Traffic Accident Reconstruction Protocol

Priority: [Urgent]

Since 2003 your country has been providing assistance toward development and economic stabilization here in Sri Lanka. Relations have gotten even closer with the invaluable help we received following the devastating tsunami in 2004. As a result, we have been able to become an important figure in the fight against terror in South-Central Asia.

As you may already know, the Sri Lanka Police Service has recently launched a new programme to update and modernize the service we provide to the public. One key area for improvement is in the Traffic Police Division. This division was established in 1953 to assist in making decisions on traffic policies and implementing them. Every currently existing station maintains a traffic branch, but the growing number of drivers on the island and our intention to build new stations demand that we immediately improve our accident investigation protocol. I am charging you with the task of compiling a new set of forensic engineering guidelines that can be used in this division.

At the moment the main focus of this task is to develop a procedure for determining if a driver has violated the speed limit. Our officers must often decide whether a person may have been speeding immediately after they are called to the scene of an accident. We need to have a step-by-step procedure for the investigator to use when he/she arrives at the scene of an accident. Please include parameters that the officer should record, as well as an easy-to-use guide on calculating an estimate of the initial speeds of the vehicles involved. In addition to this step-by-step accident investigation protocol, please provide me a cover memo describing your overall approach.

My officers will provide you a set of two abridged incident reports that are characteristic of typical accidents that we regularly investigate – please refer to our online site on April 18, 2009 for these reports. For legal reasons, sections of the reports have been omitted and the names of those involved have been replaced. In each report you will find a general description of the accident followed by more detailed information pertaining to possibly relevant parameters in the accident. In your cover memo, please discuss your conclusions regarding these two accidents and any additional incidents that my officers may forward to you (detailed analysis can be provided in an appendix).

I am confident that your team will exceed our expectations.

H. Kotakadeniya

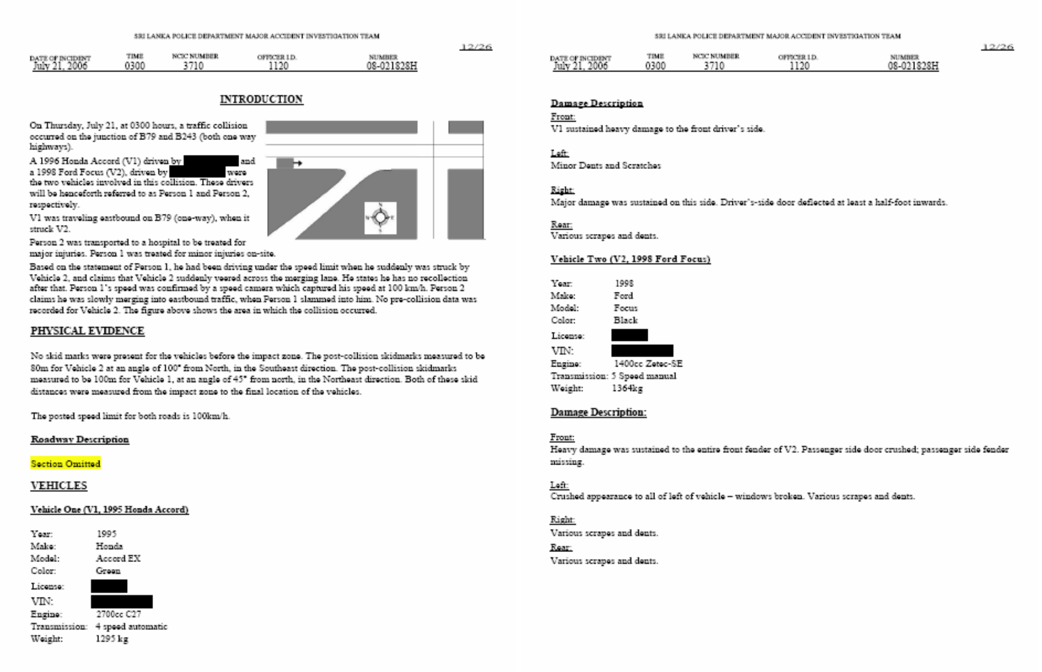
H. M. B. G. Kotakadeniya

Figure 4. Memorandum provided for VAR MEA.

The main deliverable from the VAR project was a tool for police officers to determine if vehicles were violating posted speed limits prior to collisions. With the help of Teresa Ogletree's father, who was a police officer, we were able to provide problem statements in the form of actual police reports. Students were first presented with two out of the four cases. With these two cases, they developed a generalizable model to determine which vehicle was “at fault” for each collision. Students then applied their model to two more scenarios. They could then refine their models to adequately represent the new cases if anything was previously lacking. This resulted in a model that was not only applicable to certain cases, but to crashes in general. Figure 5 shows an example of one of the cases, while all of the cases are attached in Appendix A through Appendix E

While applying their models to each case, students were required to provide a detailed explanation of all equations, assumptions, and procedures used. This allowed the MEA team to easily follow their thought processes, and to identify any common mistakes.

Most students provided a typed sheet with a method to determine pre-crash velocities. However, some students decided to use MatLab scripts or Excel spreadsheets.



SRI LANKA POLICE DEPARTMENT MAJOR ACCIDENT INVESTIGATION TEAM

12/26

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
July 21, 2006	0300	3710	1120	08-021828H

INTRODUCTION

On Thursday, July 21, at 0300 hours, a traffic collision occurred on the junction of B79 and B243 (both one way highways).

A 1996 Honda Accord (V1) driven by [redacted] and a 1998 Ford Focus (V2), driven by [redacted] were the two vehicles involved in this collision. These drivers will be henceforth referred to as Person 1 and Person 2, respectively.

V1 was traveling eastbound on B79 (one-way), when it struck V2.

Person 2 was transported to a hospital to be treated for major injuries. Person 1 was treated for minor injuries on-site.

Based on the statement of Person 1, he had been driving under the speed limit when he suddenly was struck by Vehicle 2, and claims that Vehicle 2 suddenly veered across the merging lane. He states he has no recollection after that. Person 1's speed was confirmed by a speed camera which captured his speed at 100 km/h. Person 2 claims he was slowly merging into eastbound traffic, when Person 1 slammed into him. No pre-collision data was recorded for Vehicle 2. The figure above shows the area in which the collision occurred.

PHYSICAL EVIDENCE

No skid marks were present for the vehicles before the impact zone. The post-collision skidmarks measured to be 80m for Vehicle 2 at an angle of 100° from North, in the Southeast direction. The post-collision skidmarks measured to be 100m for Vehicle 1, at an angle of 45° from north, in the Northeast direction. Both of these skid distances were measured from the impact zone to the final location of the vehicles.

The posted speed limit for both roads is 100km/h.

Roadway Description

Section Omitted

VEHICLES

Vehicle One (V1, 1995 Honda Accord)

Year:	1995
Make:	Honda
Model:	Accord EX
Color:	Green
License:	[redacted]
VIN:	[redacted]
Engine:	2700cc C27
Transmission:	4 speed automatic
Weight:	1295 kg

SRI LANKA POLICE DEPARTMENT MAJOR ACCIDENT INVESTIGATION TEAM

12/26

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
July 21, 2006	0300	3710	1120	08-021828H

Damage Description

Front:
V1 sustained heavy damage to the front driver's side.

Left:
Minor Dents and Scratches

Right:
Major damage was sustained on this side. Driver's-side door deflected at least a half-foot inwards.

Rear:
Various scrapes and dents.

Vehicle Two (V2, 1998 Ford Focus)

Year:	1998
Make:	Ford
Model:	Focus
Color:	Black
License:	[redacted]
VIN:	[redacted]
Engine:	1400cc Zetec-SE
Transmission:	5 Speed manual
Weight:	1364kg

Damage Description:

Front:
Heavy damage was sustained to the entire front fender of V2. Passenger side door crushed; passenger side fender missing.

Left:
Crushed appearance to all of left of vehicle – windows broken. Various scrapes and dents.

Right:
Various scrapes and dents.

Rear:
Various scrapes and dents.

Figure 5. One of the cases assigned for the VAR MEA.

The main purpose of the VAR project was to provide a meaningful exercise for students to use impulse-momentum and work-energy principles. One of the most common student misconceptions is applying the conservation of mechanical energy through an impact. Through the VAR project, we hoped that students would recognize that they should instead apply momentum principles to find initial velocities. As will be discussed, they seemed to have a better understanding of momentum and impact principles after completing the MEA.

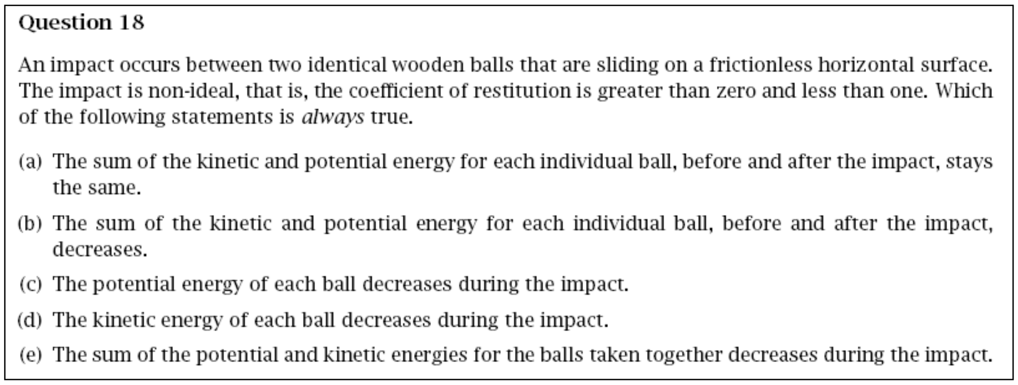
Changing student misconceptions

In order to gage the effectiveness of the VAR project, we compared the Dynamics Concepts Inventory (DCI) scores from classes that used the project versus classes that did not. The Dynamics Concepts Inventory is a set of 29 conceptual multiple choice questions related to the fundamental concepts presented in the Dynamics course (Gray 2005). The following statistics are taken from “Is There a Correlation between Conceptual Understanding and Procedural Knowledge in Introductory Dynamics.” Lora Goodwin, a member of our research team, submitted this paper to the 2009 ASEE PSW conference (Goodwin 2009). The following table displays the DCI performance of students that have been exposed to MEAs in their coursework along with those who had not.

Table 2. Total pre and post DCI scores for all MEA and non-MEA participants.

	N	Value	Pre DCI Results [%]	Post DCI Results [%]	Overall Average Normalized Gain [%]	Overall Average Percent Improvement [%]
MEA in Coursework	149	Mean	29.85	49.97	29.6	20.11
		Median	27.59	48.28		
		Standard Deviation	14.55	17.20		
No MEA's in Coursework	80	Mean	32.97	46.64	21.1	13.66
		Median	31.03	44.83		
		Standard Deviation	14.19	18.33		

As shown in Table 2, a higher normalized gain is present for students that had been assigned MEAs in their coursework. However, to highlight the effect of the VAR MEA itself, the two questions from the DCI relating to impact and momentum were studied. The questions are shown in the following figures.


Question 18

An impact occurs between two identical wooden balls that are sliding on a frictionless horizontal surface. The impact is non-ideal, that is, the coefficient of restitution is greater than zero and less than one. Which of the following statements is *always* true.

(a) The sum of the kinetic and potential energy for each individual ball, before and after the impact, stays the same.

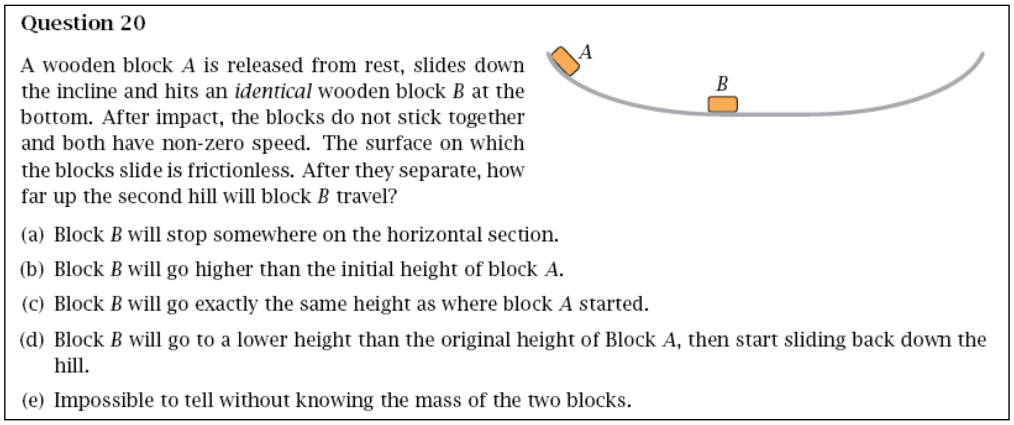
(b) The sum of the kinetic and potential energy for each individual ball, before and after the impact, decreases.

(c) The potential energy of each ball decreases during the impact.

(d) The kinetic energy of each ball decreases during the impact.

(e) The sum of the potential and kinetic energies for the balls taken together decreases during the impact.

Figure 6. Question 18 of the DCI testing students' understanding of an impact.



Question 20

A wooden block *A* is released from rest, slides down the incline and hits an *identical* wooden block *B* at the bottom. After impact, the blocks do not stick together and both have non-zero speed. The surface on which the blocks slide is frictionless. After they separate, how far up the second hill will block *B* travel?

(a) Block *B* will stop somewhere on the horizontal section.

(b) Block *B* will go higher than the initial height of block *A*.

(c) Block *B* will go exactly the same height as where block *A* started.

(d) Block *B* will go to a lower height than the original height of Block *A*, then start sliding back down the hill.

(e) Impossible to tell without knowing the mass of the two blocks.

Figure 7. Question 20 of the DCI testing students' understanding of an impact.

Table 3 highlights the performance on DCI questions 18 and 20. Students who had MEAs in their coursework had an average normalized gain of 41.1%, compared to 14.8% for students with no MEAs in their coursework. One can conclude that the MEAs did, in fact, have a significant performance on the topic covered.

Table 3. Pre and post DCI scores for MEA and non-MEA participants considering only the DCI questions directly related to MEA topic (questions 18 and 20).

	DCI Question Number	Mean DCI Pre Score [%]	Mean DCI Post Score [%]	Normalized Gain [%]	Average Normalized Gain [%]
MEA in Coursework	Q 18	26.7	45.6	25.74	41.1
	Q 20	47.6	77.2	56.48	
No MEA's in Coursework	Q 18	19.1	32.2	16.18	14.8
	Q 20	50.9	57.5	13.37	

Table 2 shows overall student performance on the entire DCI. Students with MEAs in coursework still had a higher overall average normalized gain than those without MEAs in coursework. However, when compared to the normalized gain only for questions 18 and 20 shown in Table 3, the gain is much smaller. This shows that students performed much better on the concepts that did have MEA reinforcement.

VAR Time Commitment

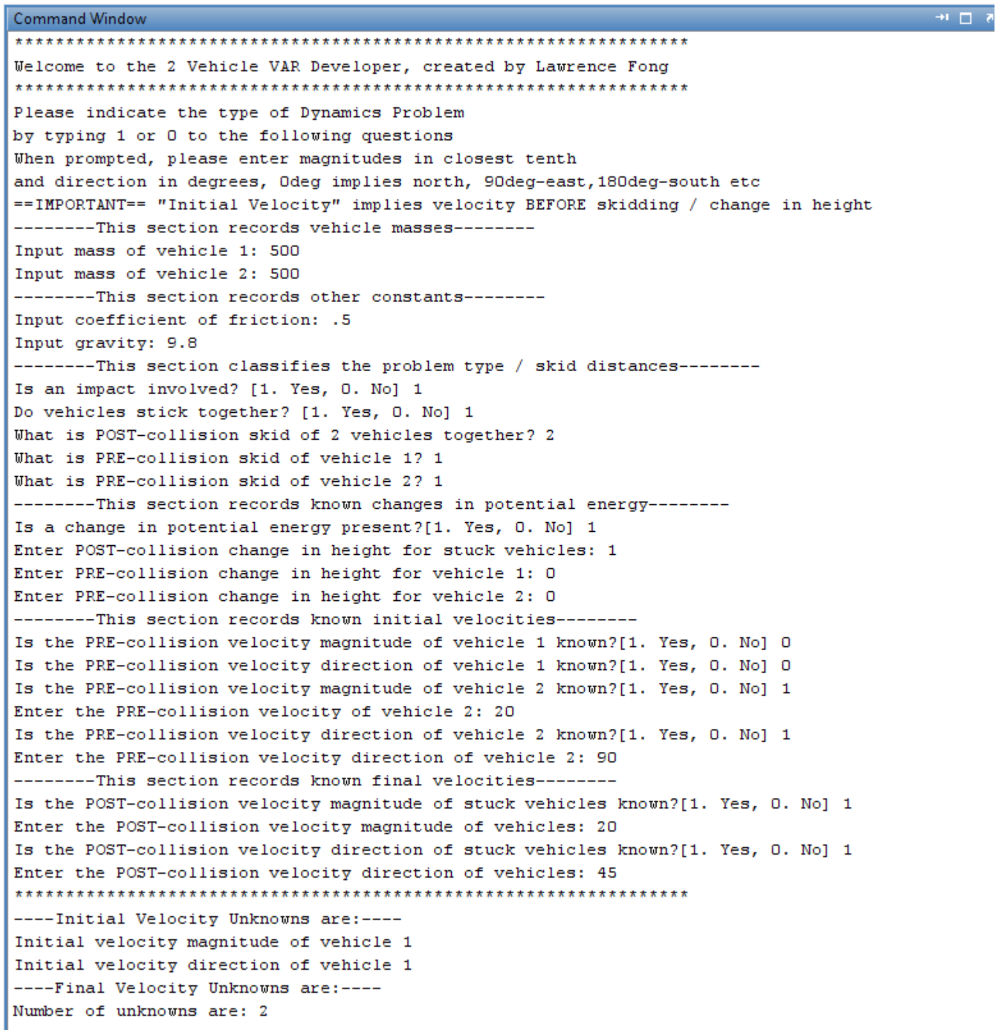
One of the greatest challenges for implementing the VAR project was the time involved for both the teachers and the students in the MEA team. Developing the problem cases required producing a new solution for every new case. Because the assignments were modified for each quarterly ME212 class, this required making a new solution set every time the VAR was assigned.

Contrary to Scott Chamberlin's "How Does the Problem Based Learning Approach Compare to the Model-Eliciting Activity Approach in Mathematics?" we found that the implementation time of the VAR MEA took significantly longer than his stated "1-2 hours required". However, Chamberlin's interpretation of the time allotted for MEAs may not be applicable to the engineering environment since engineering MEAs that we have developed were much more complex. For example, some MEAs that require only a basic statistical analysis can be conducted in less than a single class period. However, in our case, students and instructors must dedicate much more time deriving and interpreting these models. For the VAR MEA, students worked several hours outside of the allotted lecture period, and the research team spent over twenty hours grading approximately forty turn-ins.

VAR MatLab Development

In order to reduce some of this workload for instructors, I developed a MatLab code that would automatically solve for pre-collision velocities. By having this program available, we could easily change the parameters of our VAR cases and instantly have a supporting solution. It also greatly aided in the development of new cases - we could check and modify values to yield realistic solutions. The MatLab program and supporting user guide are shown in Appendix F and Appendix G . The overall goal was to have an easy-to-use program for the VAR MEA development, which could be distributed to universities that were interested in using our MEAs. Because of this, the program was revised and rewritten several times to promote ease of use.

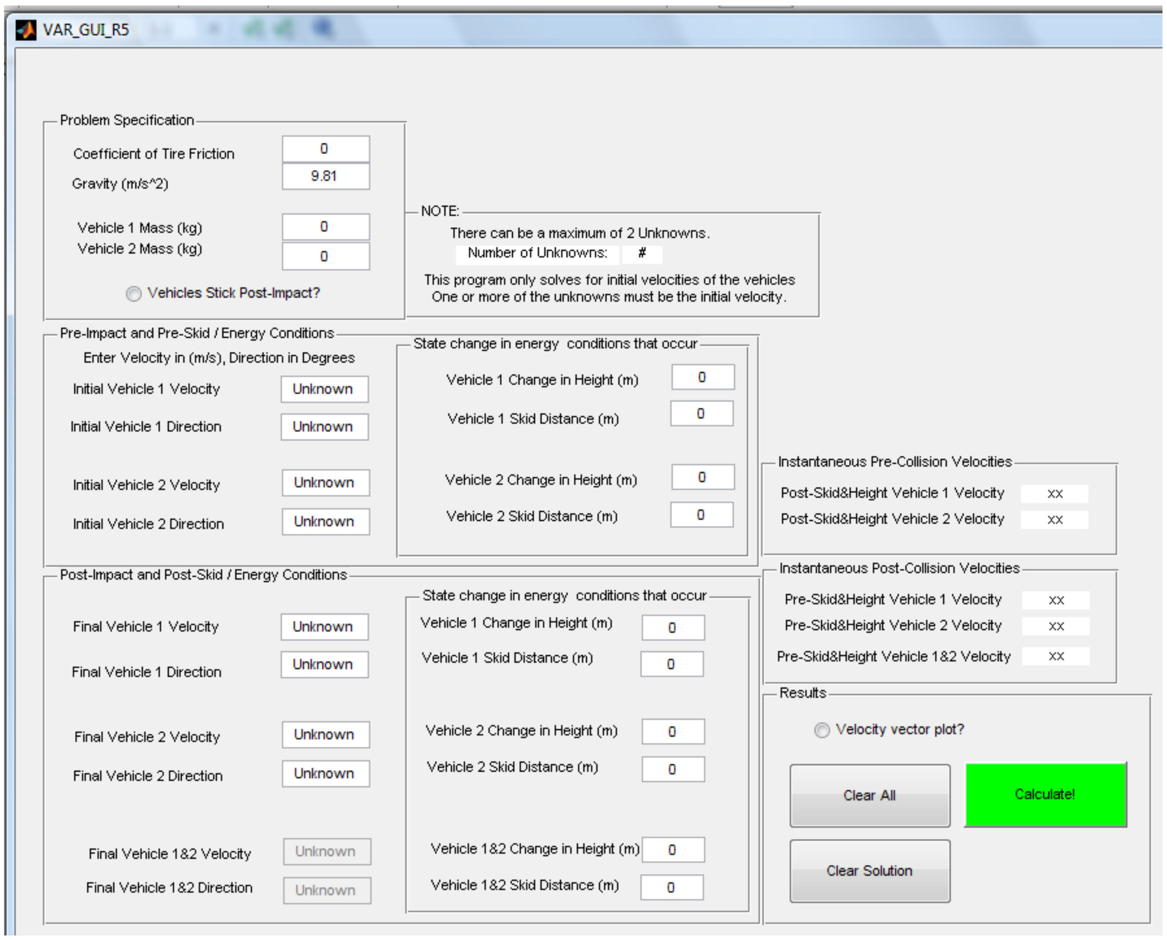
The first version of the VAR MatLab program was a line-by-line user input script. The input-window version is shown in Figure 8 below. Although functional, this line-by-line script lacked the amount of functionality I wanted for a program that would be distributed to a range of universities. It proved very cumbersome for the team when we attempted to use it to solve our own cases. Another large issue we encountered was that when we made any error in typing values in, we were unable to correct our changes – instead we had to terminate the program and reenter all the parameters again.



```
*******************************************************************
Welcome to the 2 Vehicle VAR Developer, created by Lawrence Fong
*******************************************************************
Please indicate the type of Dynamics Problem
by typing 1 or 0 to the following questions
When prompted, please enter magnitudes in closest tenth
and direction in degrees, 0deg implies north, 90deg-east,180deg-south etc
==IMPORTANT== "Initial Velocity" implies velocity BEFORE skidding / change in height
--------This section records vehicle masses--------
Input mass of vehicle 1: 500
Input mass of vehicle 2: 500
--------This section records other constants--------
Input coefficient of friction: .5
Input gravity: 9.8
--------This section classifies the problem type / skid distances--------
Is an impact involved? [1. Yes, 0. No] 1
Do vehicles stick together? [1. Yes, 0. No] 1
What is POST-collision skid of 2 vehicles together? 2
What is PRE-collision skid of vehicle 1? 1
What is PRE-collision skid of vehicle 2? 1
--------This section records known changes in potential energy--------
Is a change in potential energy present?[1. Yes, 0. No] 1
Enter POST-collision change in height for stuck vehicles: 1
Enter PRE-collision change in height for vehicle 1: 0
Enter PRE-collision change in height for vehicle 2: 0
--------This section records known initial velocities--------
Is the PRE-collision velocity magnitude of vehicle 1 known?[1. Yes, 0. No] 0
Is the PRE-collision velocity direction of vehicle 1 known?[1. Yes, 0. No] 0
Is the PRE-collision velocity magnitude of vehicle 2 known?[1. Yes, 0. No] 1
Enter the PRE-collision velocity of vehicle 2: 20
Is the PRE-collision velocity direction of vehicle 2 known?[1. Yes, 0. No] 1
Enter the PRE-collision velocity direction of vehicle 2: 90
--------This section records known final velocities--------
Is the POST-collision velocity magnitude of stuck vehicles known?[1. Yes, 0. No] 1
Enter the POST-collision velocity magnitude of vehicles: 20
Is the POST-collision velocity direction of stuck vehicles known?[1. Yes, 0. No] 1
Enter the POST-collision velocity direction of vehicles: 45
*******************************************************************
----Initial Velocity Unknowns are:----
Initial velocity magnitude of vehicle 1
Initial velocity direction of vehicle 1
----Final Velocity Unknowns are:----
Number of unknowns are: 2
```

Figure 8. Input-window for VAR development program.

Because of these issues, I decided to reprogram the script into a GUI format. Although the code itself became a bit more cluttered, a GUI was far more intuitive to use. Revision 5 of the MatLab GUI is shown in Figure 9. This program allowed testing of VAR cases much more quickly – errors could be corrected easily, and all parameters could be inputted before the calculation code executed.



Problem Specification

Coefficient of Tire Friction	0
Gravity (m/s^2)	9.81
Vehicle 1 Mass (kg)	0
Vehicle 2 Mass (kg)	0

Vehicles Stick Post-Impact?

NOTE:
There can be a maximum of 2 Unknowns.
Number of Unknowns: #
This program only solves for initial velocities of the vehicles
One or more of the unknowns must be the initial velocity.

Pre-Impact and Pre-Skid / Energy Conditions
Enter Velocity in (m/s), Direction in Degrees

Initial Vehicle 1 Velocity	Unknown
Initial Vehicle 1 Direction	Unknown
Initial Vehicle 2 Velocity	Unknown
Initial Vehicle 2 Direction	Unknown

State change in energy conditions that occur

Vehicle 1 Change in Height (m)	0
Vehicle 1 Skid Distance (m)	0
Vehicle 2 Change in Height (m)	0
Vehicle 2 Skid Distance (m)	0

Instantaneous Pre-Collision Velocities

Post-Skid&Height Vehicle 1 Velocity	xx
Post-Skid&Height Vehicle 2 Velocity	xx

Post-Impact and Post-Skid / Energy Conditions

Final Vehicle 1 Velocity	Unknown
Final Vehicle 1 Direction	Unknown
Final Vehicle 2 Velocity	Unknown
Final Vehicle 2 Direction	Unknown
Final Vehicle 1&2 Velocity	Unknown
Final Vehicle 1&2 Direction	Unknown

State change in energy conditions that occur

Vehicle 1 Change in Height (m)	0
Vehicle 1 Skid Distance (m)	0
Vehicle 2 Change in Height (m)	0
Vehicle 2 Skid Distance (m)	0
Vehicle 1&2 Change in Height (m)	0
Vehicle 1&2 Skid Distance (m)	0

Instantaneous Post-Collision Velocities

Pre-Skid&Height Vehicle 1 Velocity	xx
Pre-Skid&Height Vehicle 2 Velocity	xx
Pre-Skid&Height Vehicle 1&2 Velocity	xx

Results

Velocity vector plot?

Figure 9. MatLab GUI for VAR MEA development.

Figure 10 shows the output of the GUI program when the velocity vector plot is requested as an output. This vector plot indicates the instantaneous velocities immediately before and after an impact. In the case shown in the figure, vehicle 1 is traveling northbound, while vehicle 2 is traveling eastbound. The two vehicles collide and stick together, resulting in a post-collision velocity in the northeast direction.

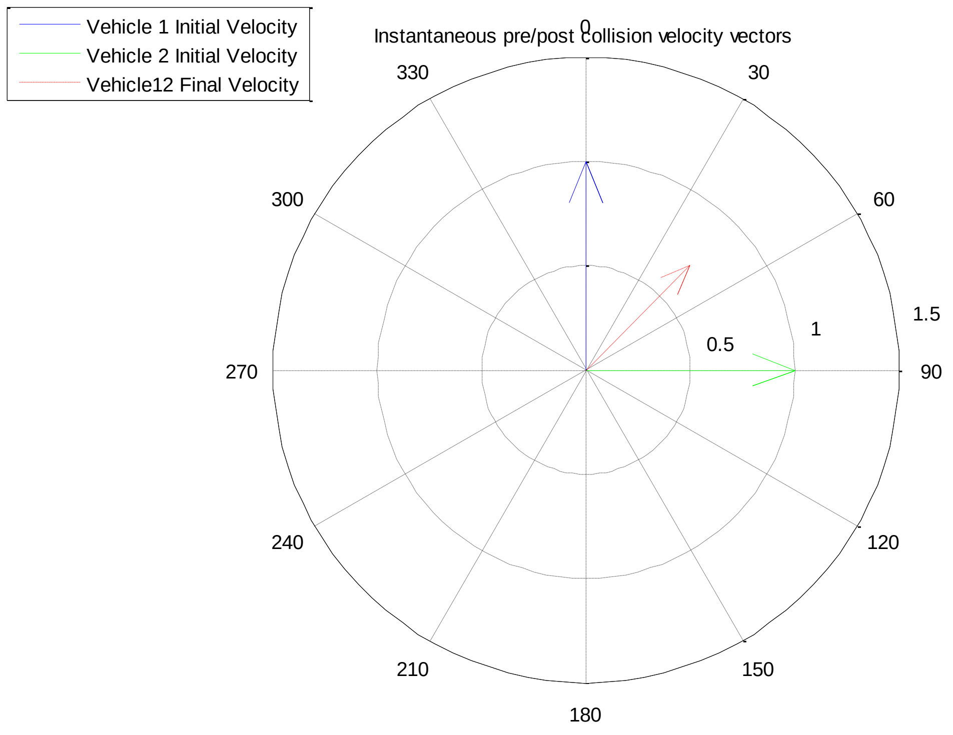


Figure 10. Velocity vector plot for instantaneous pre and post collision velocities, generated with the GUI.

VAR Cases

The VAR cases were broken down into several cases for MatLab to properly solve. MatLab has the capability of solving systems of equations using an add-in called "Symbolic Toolbox". However, I tried to avoid using any plug-ins when programming these cases so that all universities with a normal version of MatLab could use this program.

The crash scenarios were broken down into the cases shown in Table 4 and Table 5. Note that whenever possible, cases were consolidated for both post collision stick and non-stick conditions. A "stick" scenario is defined as two vehicles joining together post-collision to form a single mass – an inelastic collision. A "non-stick" scenario is defined as the two vehicles having independent masses and velocities post-collision. The equations used for solving the "stick" and "non-stick" collisions are shown in equations (1) and (2), respectively.

Table 4. Cases for 2 vehicle collisions where vehicles do not stick together post-impact.

	Pre-Collision				Post-Collision			
	V1		V2		V1		V2	
Case	Mag	Dir	Mag	Dir	Mag	Dir	Mag	Dir
A	1	1	0	0	1	1	1	1
G	1	1	0	1	1	1	1	0
D	1	1	0	1	1	1	0	1
F	1	0	0	1	1	1	1	1
C	0	1	0	1	1	1	1	1
E	1	1	0	1	1	1	1	1

Table 5. Cases for 2 vehicle collisions where vehicles stick together post-impact.

	Pre-Collision				Post-Collision	
	V1		V2		V12	
Case	Mag	Dir	Mag	Dir	Mag	Dir
A	1	1	0	0	1	1
B	1	1	0	1	0	1
G	1	1	0	1	1	0
F	1	0	0	1	1	1
C	0	1	0	1	1	1
E	1	1	0	1	1	1

$$m_1\overline{v_{1i}} + m_2\overline{v_{2i}} = m_{12}\overline{v_{12f}} \tag{1}$$

$$m_1\overline{v_{1i}} + m_2\overline{v_{2i}} = m_1\overline{v_{1f}} + m_2\overline{v_{2f}} \tag{2}$$

For all the cases, one of the unknowns was an initial velocity magnitude – as this was the most important criteria for students to determine fault in the accident scenarios. Except for case E, which has 1 unknown, every case has 2 unknowns to avoid overdefining the problem. Case E is a head-on collision, therefore only 1 unknown is allowed. Table 4 and Table 5 illustrate the known and unknown parameters of the problem indicated by “1” and “0”, respectively. For example Case A, shown in Figure 11 and Figure 12, would be a crash scenario where magnitude and direction of one vehicle’s initial velocity are unknown, while all the other pre and post collision velocities are known.

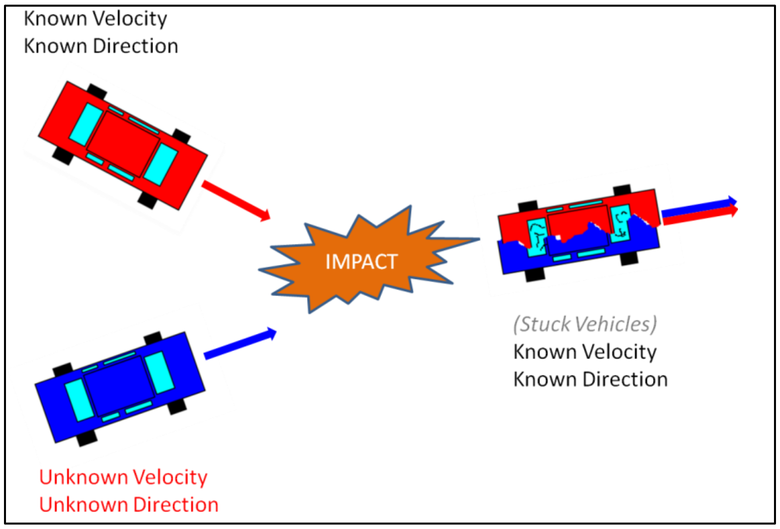


Figure 11. Case A for stick collision.

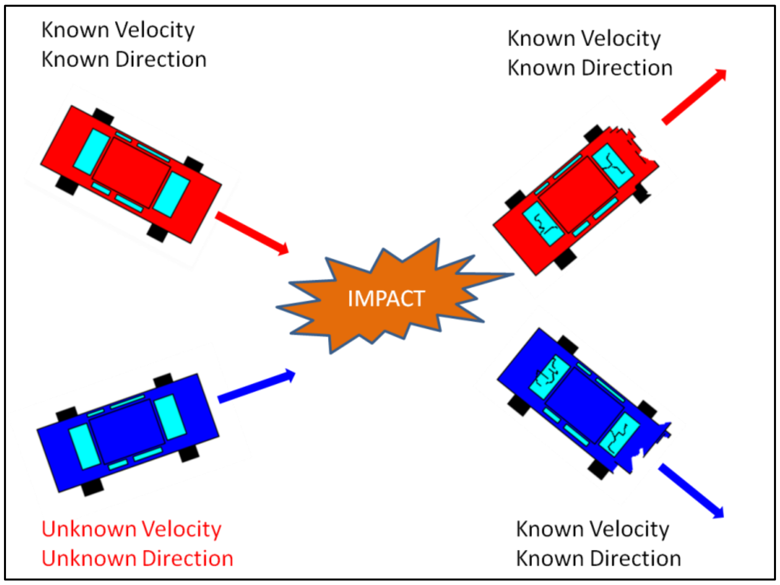
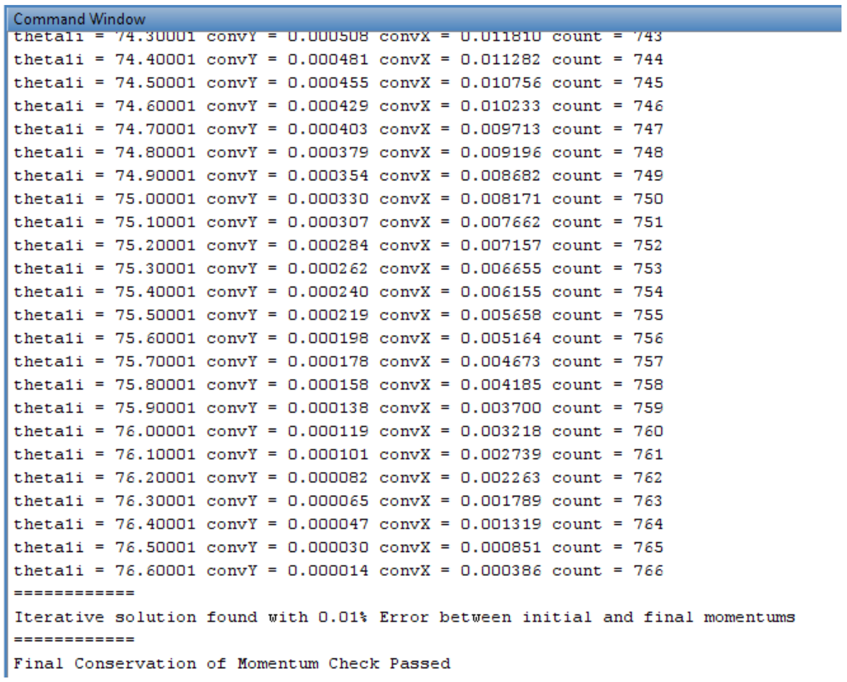


Figure 12. Case A for non-stick collision.

All of the cases, except for G and F, were solved analytically, which produced an exact result. Case A and Case E were easily solved algebraically, using equations (1) and (2).

However, cases B, C, and D were more calculation intensive, although they were still solved in algebraic form. While attempting to solve by hand, I obtained results that were algebraically correct, but had divide-by-zero errors when implementing them into the MatLab code. This could have been due to the equations not being in the most simplified form. Therefore, I utilized the symbolic toolbox to solve for the corresponding equations for these cases, in their simplest form. These equations were then implemented into the MatLab program. Note that although symbolic toolbox was used to solve for the equations, it was not used in the GUI program itself – the symbolic toolbox add-in is not required to run the program. The supported MatLab derivations are shown in Appendix H .

Cases G and F, highlighted in blue in Table 4 and Table 5, had to be solved iteratively since the unknown variable could not be isolated by itself. This iteration was done by guessing for the unknown direction, solving for the velocities, and checking conservation of momentum within a certain percent error. Because of this, the produced solution was an approximate answer. The initial convergence criteria for initial and final momentum convergence in the x and y direction was 0.01%. However, if the solution did not properly converge, the iteration code relaxed the criteria in two stages. The first stage "relaxed" convergence criteria used 1% between the final and initial momentum, in both directions. The second stage used 2%. If neither of these criteria were met, the code exited out, producing an error.



```
Command Window
theta1i = 74.30001 convY = 0.000508 convX = 0.011810 count = 743
theta1i = 74.40001 convY = 0.000481 convX = 0.011282 count = 744
theta1i = 74.50001 convY = 0.000455 convX = 0.010756 count = 745
theta1i = 74.60001 convY = 0.000429 convX = 0.010233 count = 746
theta1i = 74.70001 convY = 0.000403 convX = 0.009713 count = 747
theta1i = 74.80001 convY = 0.000379 convX = 0.009196 count = 748
theta1i = 74.90001 convY = 0.000354 convX = 0.008682 count = 749
theta1i = 75.00001 convY = 0.000330 convX = 0.008171 count = 750
theta1i = 75.10001 convY = 0.000307 convX = 0.007662 count = 751
theta1i = 75.20001 convY = 0.000284 convX = 0.007157 count = 752
theta1i = 75.30001 convY = 0.000262 convX = 0.006655 count = 753
theta1i = 75.40001 convY = 0.000240 convX = 0.006155 count = 754
theta1i = 75.50001 convY = 0.000219 convX = 0.005658 count = 755
theta1i = 75.60001 convY = 0.000198 convX = 0.005164 count = 756
theta1i = 75.70001 convY = 0.000178 convX = 0.004673 count = 757
theta1i = 75.80001 convY = 0.000158 convX = 0.004185 count = 758
theta1i = 75.90001 convY = 0.000138 convX = 0.003700 count = 759
theta1i = 76.00001 convY = 0.000119 convX = 0.003218 count = 760
theta1i = 76.10001 convY = 0.000101 convX = 0.002739 count = 761
theta1i = 76.20001 convY = 0.000082 convX = 0.002263 count = 762
theta1i = 76.30001 convY = 0.000065 convX = 0.001789 count = 763
theta1i = 76.40001 convY = 0.000047 convX = 0.001319 count = 764
theta1i = 76.50001 convY = 0.000030 convX = 0.000851 count = 765
theta1i = 76.60001 convY = 0.000014 convX = 0.000386 count = 766
============
Iterative solution found with 0.01% Error between initial and final momentums
============
Final Conservation of Momentum Check Passed
```

Figure 13. VAR GUI iteration for Case F. convY and convX indicate the percent difference between initial and final momentum magnitudes, in the Y and X direction.

VAR GUI Summary and Recommendations

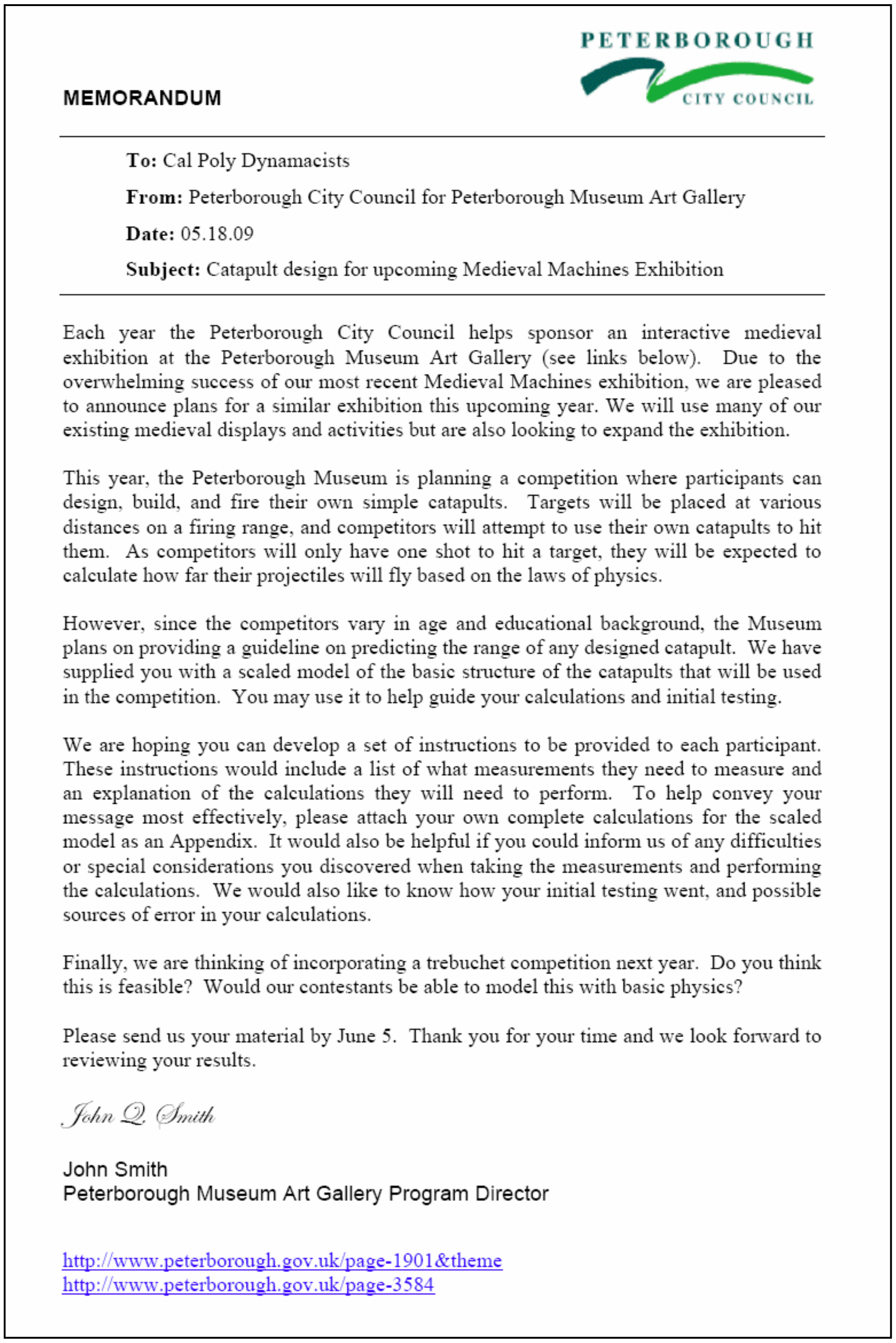
The VAR GUI was used to solve many test cases as well as various momentum problems from the ME212 textbook. So far, it has properly solved for all supported cases, however as with all software, some bugs will likely be discovered when it is put to repeated use. The GUI performs all necessary calculations, and does a final check for conservation of momentum in the x and y directions. If conservation of momentum is not passed, the code will error out with an appropriate message, which will greatly speed up troubleshooting in the future.

In retrospect, the code could be greatly simplified if all the cases were solved iteratively. However, this would result in a much greater computation time, and all the solutions would be only approximate answers, rather than analytical solutions. Overall, the GUI is a convenient tool to develop and check VAR scenarios. We intend to distribute it to other universities that are using our VAR MEAs, so that they can also reduce the workload on their instructors.

Catapult

The Catapult MEA was introduced in some sections of ME212, ‘Dynamics’ after the concepts of work-energy and angular velocity/acceleration were introduced in lecture. It was also implemented in some sections of ME326, “Intermediate Dynamics”, although not yet formulated into an appropriate MEA format. Professors Dr. Brian Self, Dr. Jim Widmann, and Dr. Peter Schuster have successfully implemented this project into ME212, and Dr. Self has used this project in ME326.

Once again, the MEA was placed in a professional client-driven setting. The memo that was presented to students in ME212 is shown in Figure 14. The “client” for this MEA was the Petersborough Museum, who needed a set of guidelines for predicting the range of projectiles fired from simple catapults. Students were then supplied with a “scaled-model” of the catapult, shown in Figure 15, in order to assist with their analysis.



PETERBOROUGH CITY COUNCIL

MEMORANDUM

To: Cal Poly Dynamacists

From: Peterborough City Council for Peterborough Museum Art Gallery

Date: 05.18.09

Subject: Catapult design for upcoming Medieval Machines Exhibition

Each year the Peterborough City Council helps sponsor an interactive medieval exhibition at the Peterborough Museum Art Gallery (see links below). Due to the overwhelming success of our most recent Medieval Machines exhibition, we are pleased to announce plans for a similar exhibition this upcoming year. We will use many of our existing medieval displays and activities but are also looking to expand the exhibition.

This year, the Peterborough Museum is planning a competition where participants can design, build, and fire their own simple catapults. Targets will be placed at various distances on a firing range, and competitors will attempt to use their own catapults to hit them. As competitors will only have one shot to hit a target, they will be expected to calculate how far their projectiles will fly based on the laws of physics.

However, since the competitors vary in age and educational background, the Museum plans on providing a guideline on predicting the range of any designed catapult. We have supplied you with a scaled model of the basic structure of the catapults that will be used in the competition. You may use it to help guide your calculations and initial testing.

We are hoping you can develop a set of instructions to be provided to each participant. These instructions would include a list of what measurements they need to measure and an explanation of the calculations they will need to perform. To help convey your message most effectively, please attach your own complete calculations for the scaled model as an Appendix. It would also be helpful if you could inform us of any difficulties or special considerations you discovered when taking the measurements and performing the calculations. We would also like to know how your initial testing went, and possible sources of error in your calculations.

Finally, we are thinking of incorporating a trebuchet competition next year. Do you think this is feasible? Would our contestants be able to model this with basic physics?

Please send us your material by June 5. Thank you for your time and we look forward to reviewing your results.

John Q. Smith

John Smith
Peterborough Museum Art Gallery Program Director

http://www.peterborough.gov.uk/page-1901&theme
http://www.peterborough.gov.uk/page-3584

Figure 14. Catapult MEA assigned to ME212 classes.

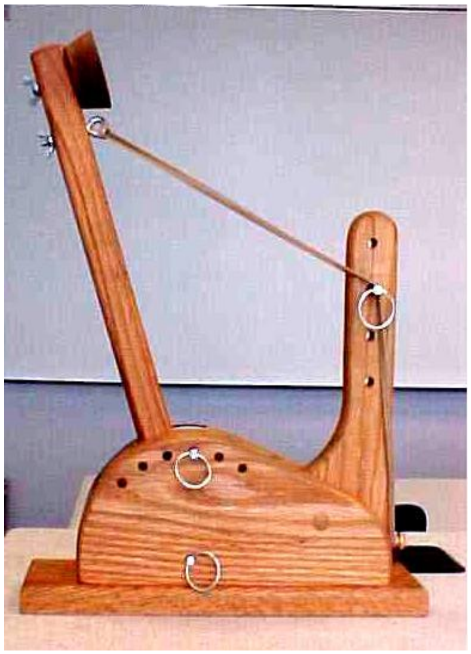


Figure 15. Catapult provided to students for analysis.

Along with the supplied scaled catapult, they were also given rubber bands, rulers, weights, and a scale. With these tools, they were expected to determine all the parameters necessary to model the catapult. In contrast to a typical textbook problem, where all the required values are already explicitly stated, this MEA required students to apply their analytical skills to actually determine what information was needed. Some of these parameters included the dimensions of the catapult arm, dimension from pivot to ammo cup, and the height of the rubber band pin. With these parameters, they had to determine the moment of inertia of the catapult arm, inclusive of the ammo cup and egg. Based on their engineering knowledge, some students made assumptions of point-masses for the egg and cup, and slender rod behavior for the catapult arm. Another important aspect was the behavior of the rubber band. Some students assumed linear behavior, using an average spring constant for their theoretical model. Other students used a curve-fit to the force-displacement data to account for any nonlinearities in rubber band stiffness, as shown in Figure 16. We wanted this open-ended aspect to stimulate the kind of critical thinking lacking in many textbook problems.

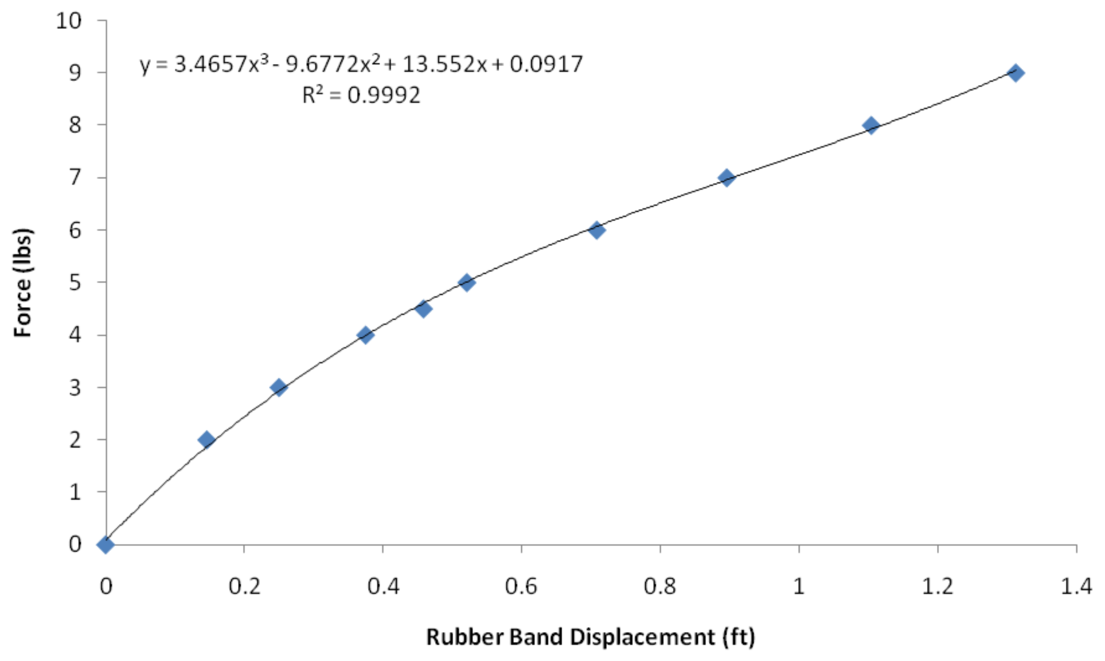


Figure 16. Force versus displacement curve for rubber band.

As their deliverables, students provided a model to predict the range of catapults in general, as well as applying their model specifically to the scaled catapult that was provided. They were required to develop a model using hand calculations that would be applicable to conditions that would be specified later - during launch day. These conditions were: stopper pin angle and pull-back angle, illustrated in Figure 17. Students should have realized that both the trajectory and distance traveled were a function of these two variables.

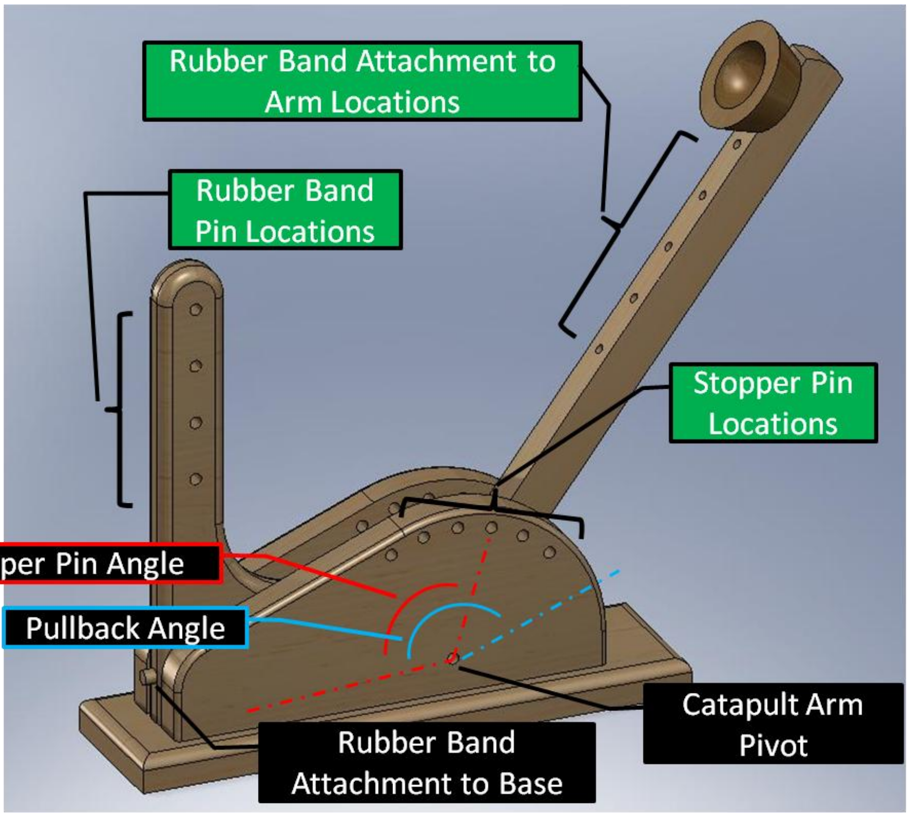

Rubber Band Attachment to Arm Locations
Rubber Band Pin Locations
Stopper Pin Locations
Stopper Pin Angle
Pullback Angle
Catapult Arm Pivot
Rubber Band Attachment to Base

Figure 17. Diagram of catapult locations.

A target was placed in front of the catapult during launch day, and a required pin stopper angle was specified. Students then adjusted the pullback angle of the catapult - based on their model - in order to hit the target. They were able to choose their own rubber band attachment and rubber band pin locations. Judging from where their egg landed, they were able to see where their calculations may have gone awry, which was a significant application of the self-assessment principle. They could then go back and rework their calculations to match up with the physical results. In addition to using their model on launch day, students completed a follow-up homework assignment which also utilized their model. This homework assignment involved finding the force on the stopper pin using impulse-momentum, as well as the force on the pivot pin using the sum of forces and moments. This allowed students to connect an additional concept from lecture to their model.

Instrumentation

Although launching the catapult was already a great way of providing validation to students' analysis, we wanted to provide further experimental data. In order to expand upon the "reality" and "self-assessment" principles of this MEA, we outlined parameters that we wanted to measure using instrumentation. By providing students with experimental data, they would be able to validate their theoretical results with physical data. This real-time data would ideally be taken by the students during launch day, where they could visually see the trajectory of the egg. Students could then compare the visual results, the experimental data, and their theoretical results. The parameters that we wanted to measure are:

- Angular Velocity
- Angular Position
- Angular Acceleration
- Axial Stress
- Force at Stopper Pin

To obtain this experimental data, we used a two-axis accelerometer and four strain gages, in conjunction with a data acquisition system. The final equipment list (after several design iterations) is listed below:

- 1x ADXL-278 ±50g Dual-Axis iMEMS Accelerometer
- 1x 5V Voltage Supply (inclusive of hardware noise filtering)
  - 1x 5V Voltage Regulator
  - 2x 10µF capacitor
  - 1x 0.1µF capacitor
  - 1x 0.01µF capacitor
- 4x CEA-06-240UZ-120 Vishay Strain Gages

- 1x NI-cRIO-9014 Real-Time Controller
- 1x NI-cRIO-9101 4-Slot, 1M Gate CompactRIO Embedded Chassis
- 1x NI-9237 Simultaneous Bridge Module
- 1x NI-9205 Analog Input Module
- 1x NI-9949 NI 9949 RJ-50 to Screw Terminal Adaptor (Strain Bridge)
- 2x 120 Ω Vishay 5-120-01 Precision Resistors

One of the most critical design considerations was the required setup time for gathering data. While proctoring ME212 students during launch day, I was *barely* able to squeeze all of the student teams' launches into the 50 minute period. I realized that essentially no time would be allotted to set up the instrumentation. Therefore, all of the following instrumentation is designed to record the data with a click of a button in LabView, with no setup time in between. Students can then use the post-processing code to analyze their results.

All of the testing in this section was conducted with one rubber band on the catapult, and no attached egg or other projectile. The reason for this was because many trials were to be conducted in the graduate lab. Launching projectiles could damage other equipment in the lab, and based on my previous experience, two rubber bands could damage the catapult arm, as shown in Figure 18.

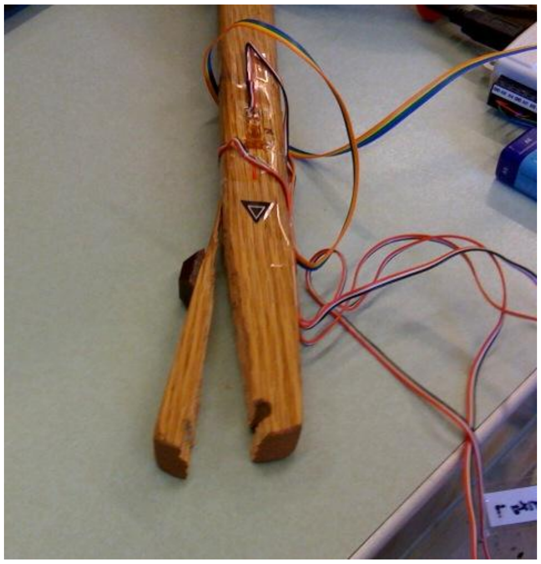


Figure 18. Broken catapult arm after some initial trials with two rubber bands.

Angular Position, Velocity, and Acceleration

Several techniques to measure angular velocity and acceleration were considered before ultimately arriving at a dual-axis accelerometer. We considered using either a rotary encoder or rotary potentiometer to measure the position of the arm as a function of time. These, however, would require a rigid attachment to both the catapult arm as well as the base. This would require significant machining and would be potentially expensive.

Our solution to measure both angular velocity and acceleration was to use an ADXL-278 accelerometer. This accelerometer was low-cost, measured acceleration in two axes, and required a rigid attachment to only the catapult arm. The full specifications of the accelerometer are provided in Appendix L . The ADXL-278 was oriented to measure both normal acceleration and tangential acceleration of the catapult arm, as shown in Figure 19. From these two accelerations, we could then directly calculate the angular velocity and acceleration of the catapult using the following relations.

$$\mathbf{a}_t = \boldsymbol{\alpha} \times \mathbf{r} \tag{3}$$

$$\mathbf{a}_n = \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}) \tag{4}$$

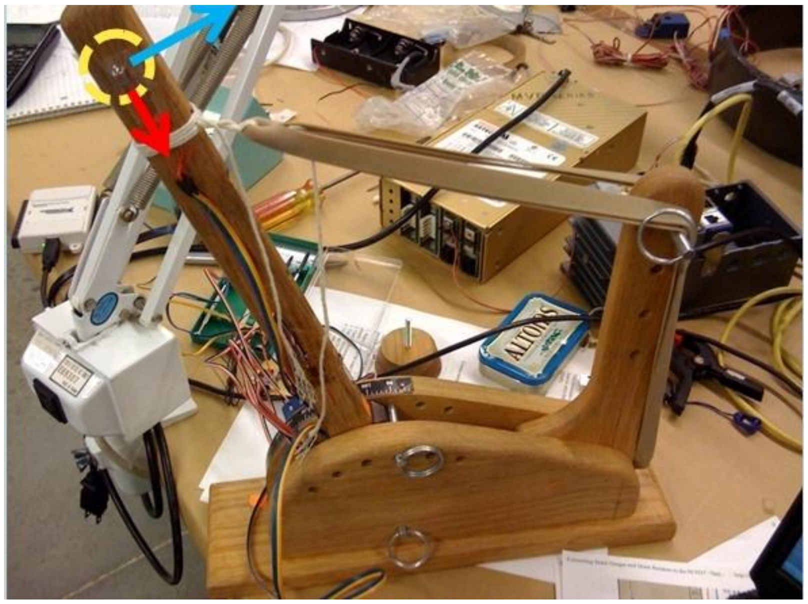


Figure 19. Catapult with accelerometer. The accelerometer is highlighted in yellow. The blue and red arrows represent the directions of tangential and normal acceleration, respectively.

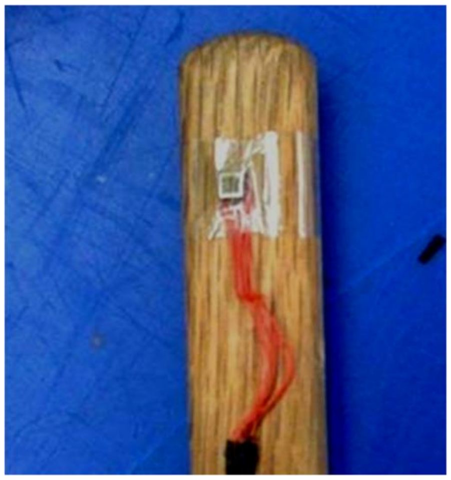


Figure 20. Close-up of accelerometer on catapult arm.

For noise reduction in the power supply, we utilized the circuit shown in Figure 21 to power the accelerometer and for some hardware signal conditioning. The circuit is a combination of a 5V voltage regulator (with capacitors for noise reduction) and a grounded 0.01μF capacitor at each of the two outputs of the accelerometer. The outputs of both axes are then read into a data acquisition unit.

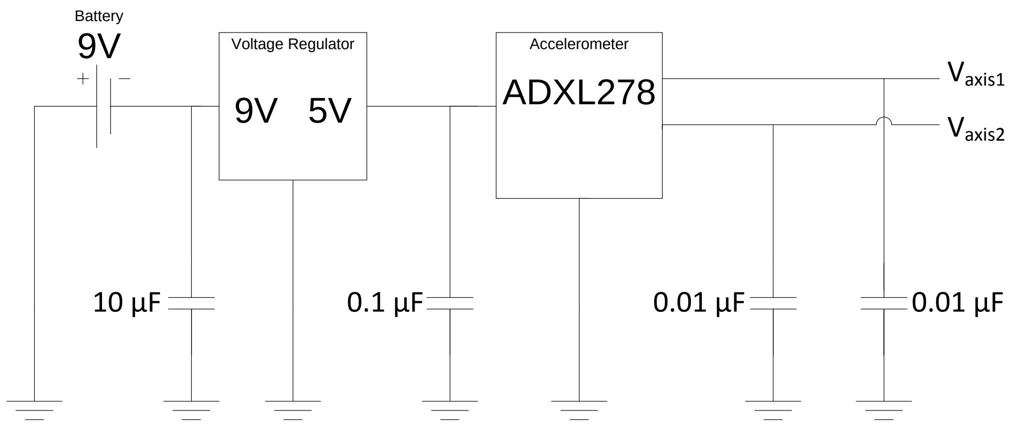


Figure 21. Circuit diagram for 5V voltage regulator & hardware accelerometer signal filtering.

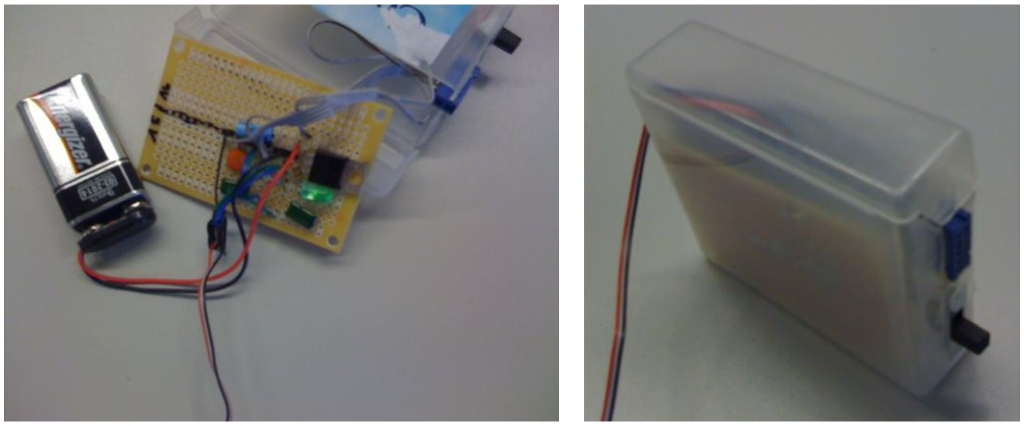


Figure 22. Voltage regulator internals and packaging.

In order to read the analog signal from the accelerometer into the computer, we needed a data acquisition system with an appropriate sample rate. Our first iteration utilized a NI USB-6008. This DAQ had a maximum sampling rate of 5000Hz, which was more than fast enough for our catapult duration of less than 60 milliseconds. The accelerometer was then tested for repeatability, shown in Figure 23.

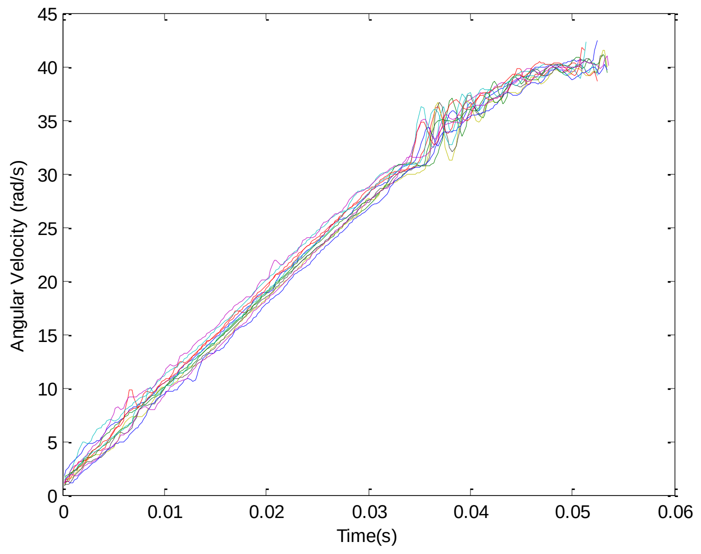


Figure 23. Angular velocity data for multiple runs, using NI-6008.

One of the major drawbacks of using an accelerometer versus an encoder was that the position of the catapult arm had to be calculated by integrating the angular velocity. While this was initially a concern, Figure 24 shows the results of a numerical integration of the angular velocity to yield the corresponding position. A summary of these results is provided in Table 6.

Table 6. Summary of 12 repeatability tests for pullback angle of 72°

	Experimental Parameter	Average Numerical Integration Output	Percent Difference	Standard Deviation
Pullback angle – stopper angle	72°	71.2°	1 %	1.025°

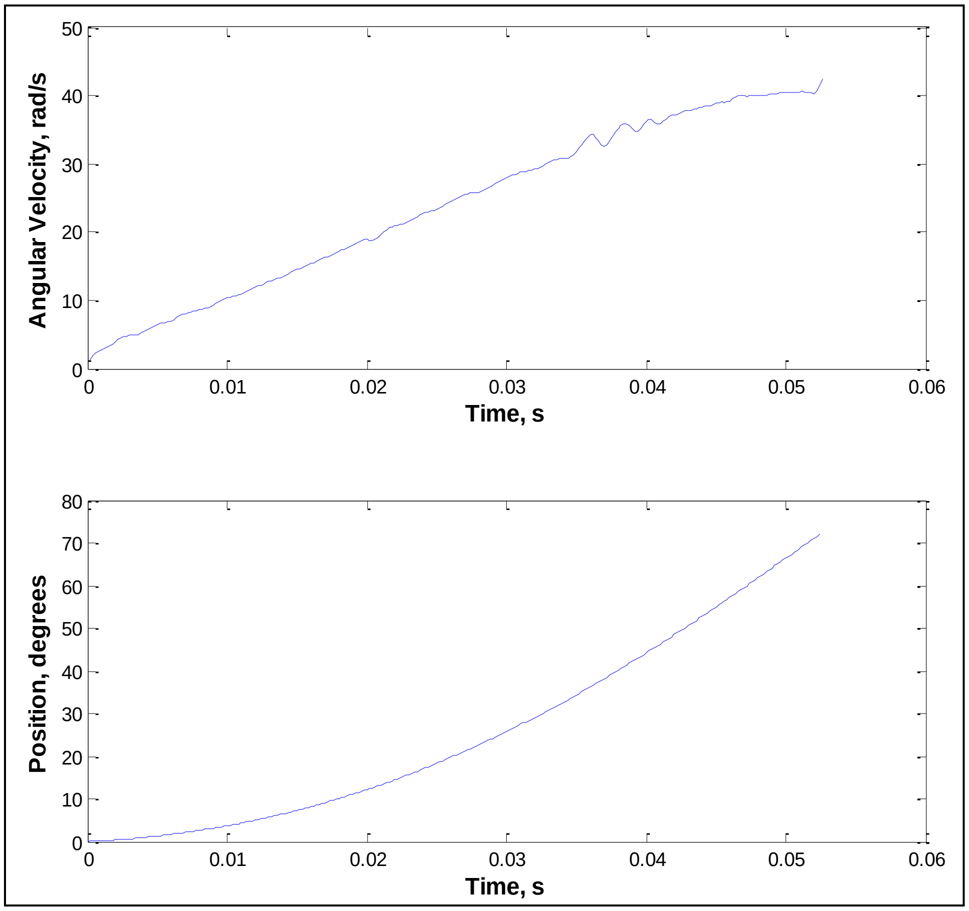


Figure 24. Experimental results for angular velocity and position. Position was calculated from numerical integration of velocity.

Switching to CompactRIO

Although the USB-NI-6008 was adequate for accelerometer measurements alone, a problem arose when we tried to interface it with a strain measurement. Since the output voltage from any strain gage measurement was well under the minimum voltage that could be read by the USB-NI-6008, a separate module had to be used for the strain voltage. We had originally planned to purchase a USB adapter for the C-Series NI-9237 strain module – in that manner we could hook up both the NI-6008 and the NI-9237 via USB ports, and sample from each in LabView. However, a “lag time” would be present between readings of the two USB devices, which would cause trouble in synchronizing the two signals. This would be less of a problem if the event time was long – however since the catapult motion occurs in less than 60 milliseconds, this became a concern.

Rather than purchasing an adapter that may not have worked, we instead turned our attention to a CompactRIO (National Instruments, Austin, TX), which was readily available for me to use. Typically, the CompactRIO modules are used to record data without the need to be hooked up to a computer. However, in our case, we wanted the data to be displayed on the computer screen as the event was occurring. This required the usage of "scan mode" on the CompactRIO, which omits the requirement of any FPGA programming. The configuration of the CompactRIO required us to use the "Using CompactRIO Scan Mode with Unsupported Backplanes", stated on the NI website.

Switching to this CompactRIO required changing the voltage module used to read acceleration from the USB-NI-6008 to the NI-9205 Analog Input Module. This was not a problem, however, the maximum sampling rate was reduced from 5000Hz to 1000Hz. Even so, as shown in Figure 25, an adequate amount of samples was obtained using this reduced sampling rate.

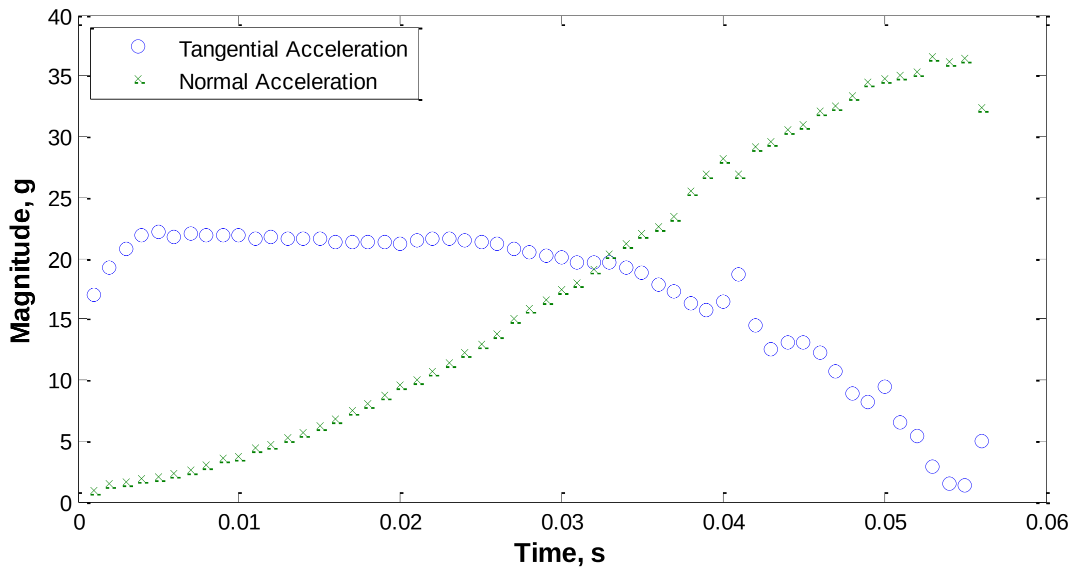


Figure 25. Trial for testing reduced sampling rate of 1000Hz, using CompactRIO and NI-9205.

The top plot of Figure 26 compares the values of experimental angular velocity with the angular velocity obtained by a rectangular numerical integration of the experimental angular acceleration. The offset between the two values was most likely caused by the orientation of the accelerometer. The bottom plot of Figure 26 compares the values of experimental angular acceleration with the angular acceleration obtained by deriving the experimental angular velocity. Because of the noise in the normal acceleration direction, from which the angular velocity was calculated, calculating the angular acceleration using the two-point backwards difference and four-point central difference methods of numerical differentiation were also noisy. However, the derived results fluctuated about the experimentally obtained value. Therefore, this could potentially be a good method of checking the orientation of the accelerometer.

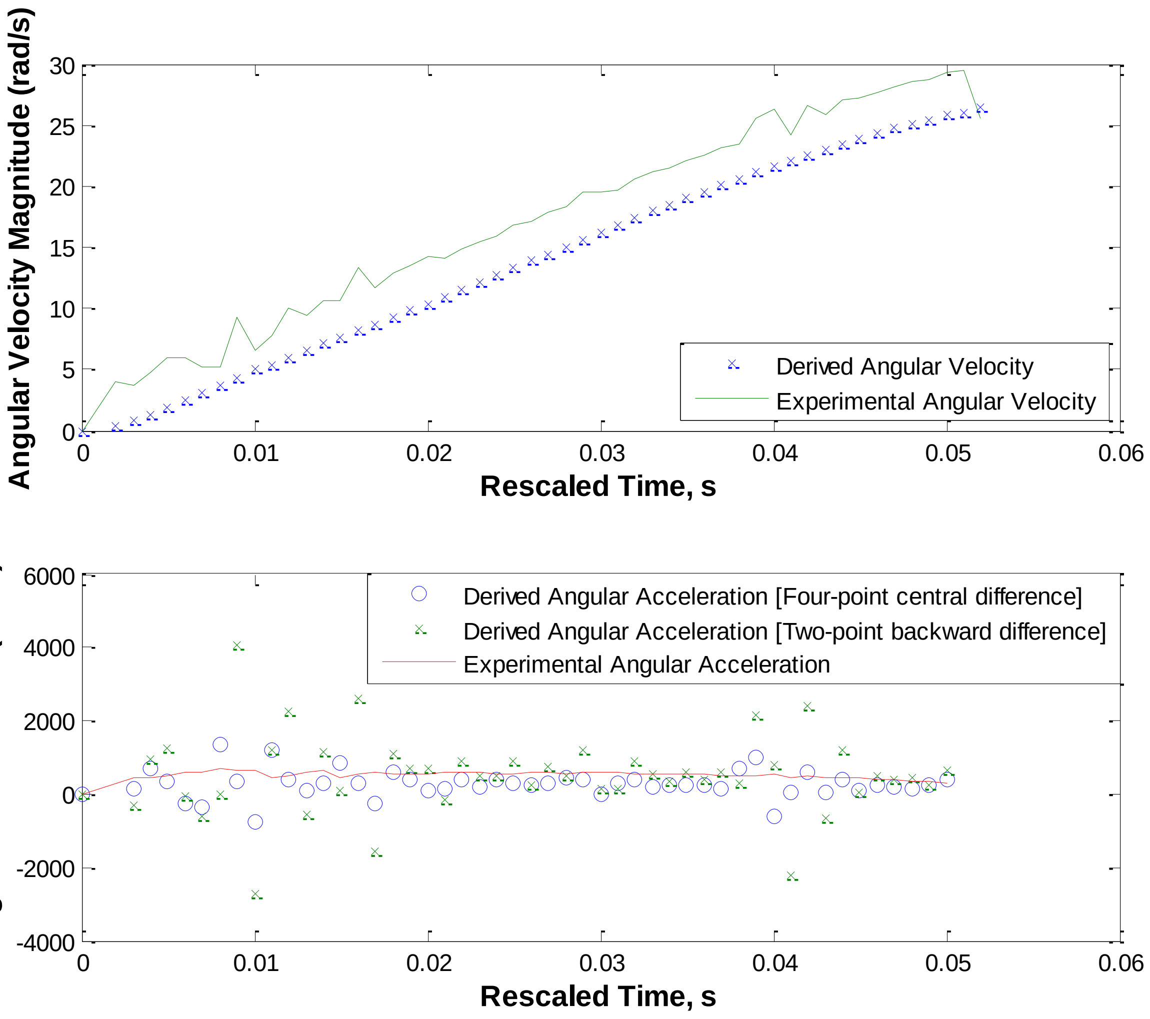


Figure 26. (Top) Comparison between experimental angular velocity and integrated angular velocity from experimental angular acceleration. (Bottom) Comparison between experimental angular acceleration and derived angular acceleration using experimental angular velocity.

Axial Stress and Force at Stopper Pin

In order to capture the force at the stopper pin and the axial stress, we used strain gages to measure the bending and axial strain of a catapult arm location during launch. The bending strain was caused by the tangential force from the rubber band, which in turn accelerated the catapult arm. A large bending strain was also present during the impact time of the catapult arm with the stopper pin. The axial strain was caused by a combination of the axial force from the catapult arm as well as the normal acceleration of the effective center of mass above the mounted gage. An FBD and MAD of the catapult arm are shown in Figure 27.

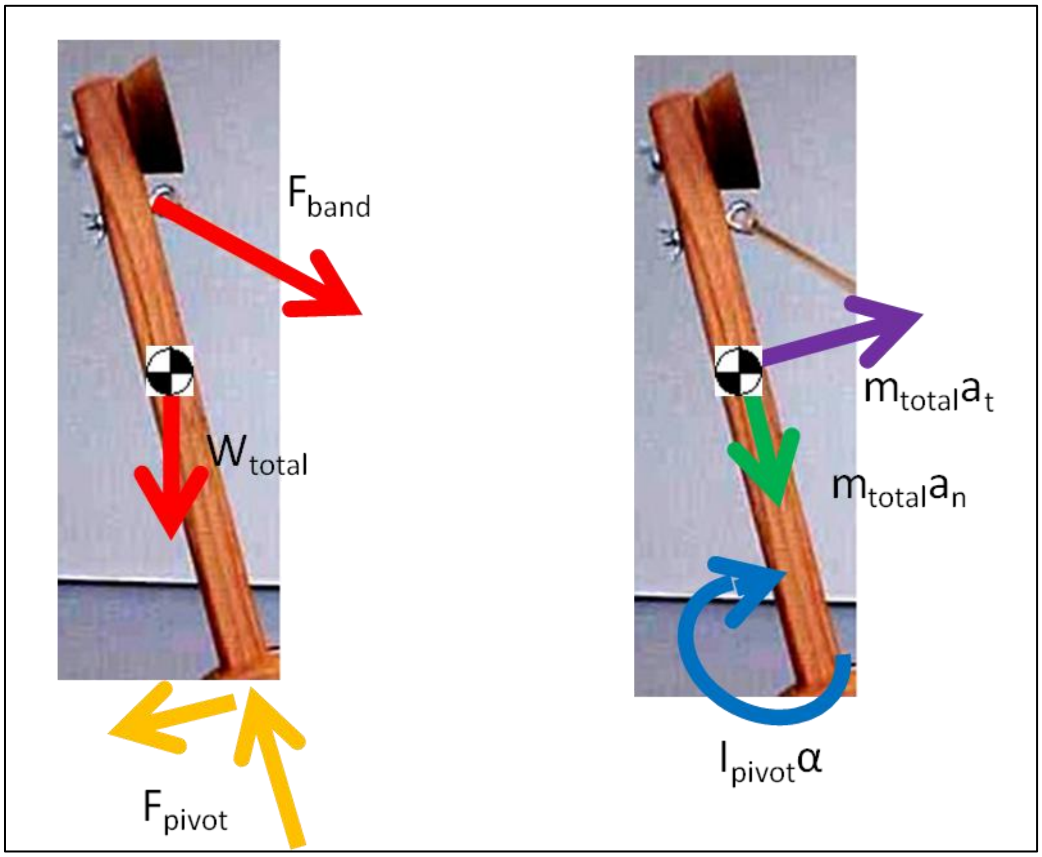


Figure 27. FBD and MAD of catapult arm.

The strain gages were assembled into a “full” Wheatstone bridge as shown in Figure 28. For axial strain, we positioned strain gages at (1) and (3), and placed precision resistors at (2) and (4). This configuration resulted in measuring axial strain only – bending effects were cancelled out. For bending strain, we positioned strain gages at (1) and (4), and used the internal completion NI-9949 resistors at (2) and (3). This configuration resulted in measuring bending strain only – axial effects were cancelled out. The bridge setup and associated equations were taken from James W. Dally’s Instrumentation for Engineering Measurements, 2nd edition. During any applied load, the resultant axial strain and moment strain could be determined by the following reduced equation:

$$\frac{V_o}{V_{ex}} = \frac{S_g \in}{2} \tag{5}$$

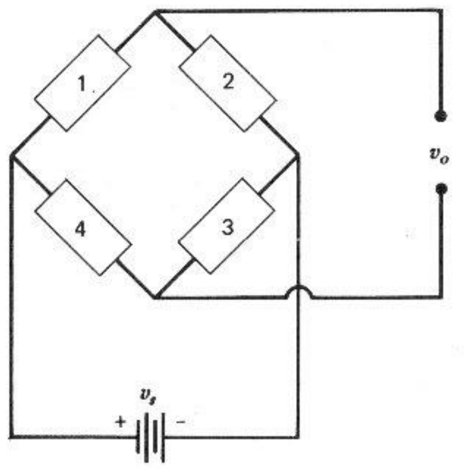


Figure 28. "Full" Wheatstone bridge configuration for strain gages. <u>For axial strain</u>: (1) and (3) represent mounted gages, while (2) and (4) represent external precision resistors. <u>For bending strain</u>: (1) and (4) represent mounted gages, while (2) and (3) represent internal precision resistors.

The strain gages were mounted using m-line AE10 epoxy, as per the instructions in the Vishay manual (Vishay Micro-Measurements 2005). Figure 29 shows the overnight curing of the strain gages under pressure from clamps. An illustration of the strain gage locations is provided in Figure 30 and the actual strain gages are shown in Figure 31.

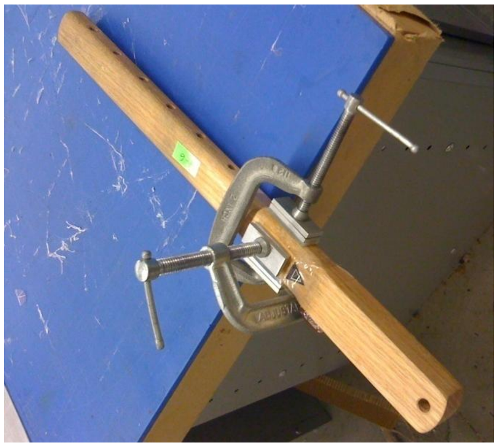


Figure 29. Strain gages curing under pressure from c-clamps.

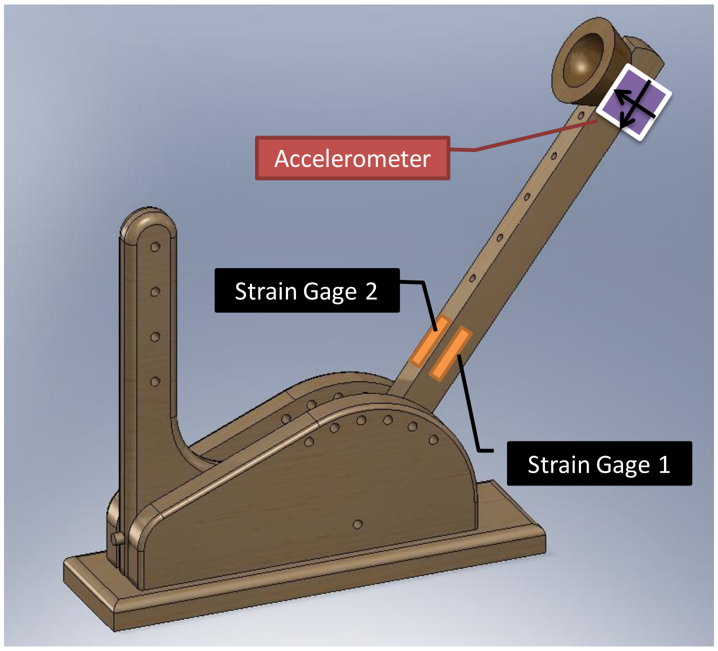


Figure 30. Illustration of catapult with associated instrumentation. Strain gage 3 (not visible) is mounted directly opposite of strain gage 1. Strain gage 4 (not visible) is mounted directly opposite of strain gage 2.

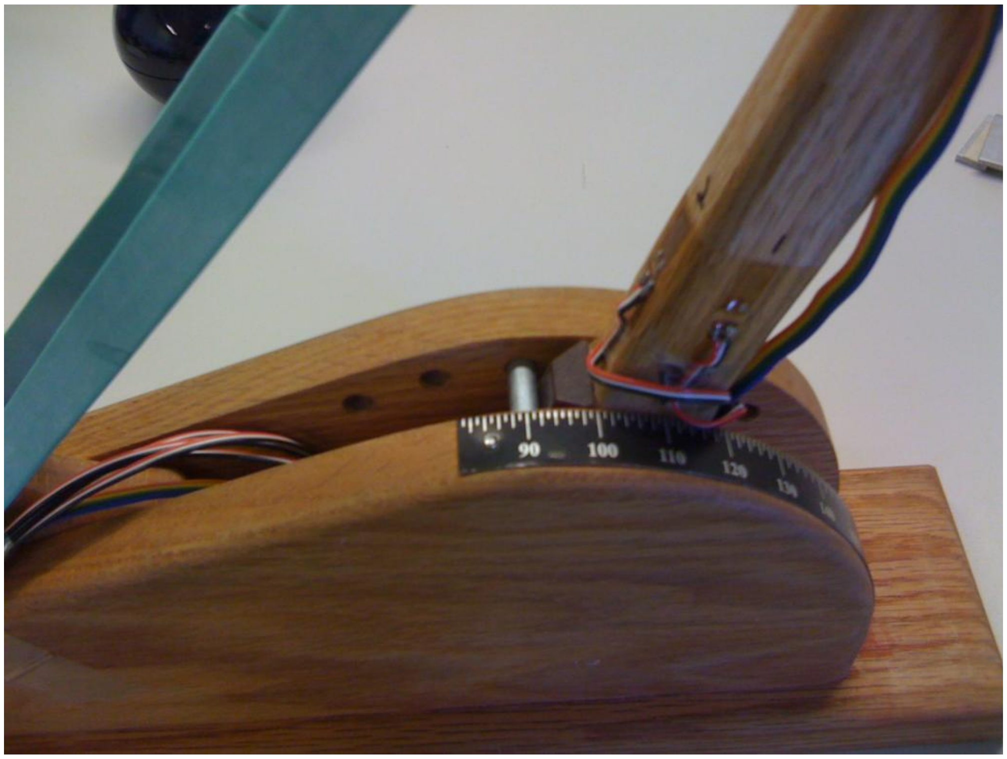


Figure 31. Catapult with mounted strain gages.

Because we were utilizing the strain gages as force/stress transducers, we needed to find the modulus of elasticity of the wood. By experimentally determining this modulus of elasticity, we could then relate the recorded strain to stress using Hooke's Law:

$$\sigma = E \in \tag{6}$$

The modulus of elasticity was found by loading the catapult arm axially, and recording resultant strains caused by static loading of weights. This was accomplished by hanging the catapult arm from a ladder, and hanging combinations of 2lb and 10lb weights, as shown in Figure 33. The associated strains were measured using a P3 Strain Indicator and Recorder (Vishay).

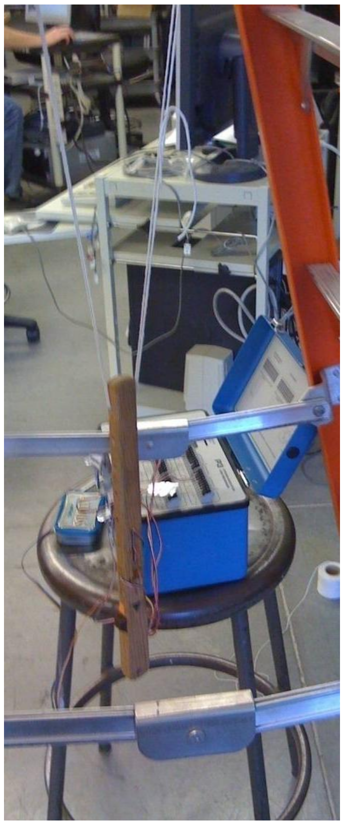


Figure 32. Static loading of catapult arm to determine modulus of elasticity.

The results from the modulus of elasticity test are shown in and Table 7. Wood typically has a nonlinear behavior when not in the direction of the grain structure, but fortunately the grains were oriented in the direction of our applied force. The experimental value for the elastic modulus of the catapult arm were very close to the published value for oak wood of 1.49 Msi (Smithsonian Institution 1969, 246), along the direction of the grain.

Table 7. Comparison of experimental published value for catapult arm and published elastic modulus for oak.

	Experimental	Published	% Difference
Elastic Modulus (Msi)	1.53	1.49	-2.0%

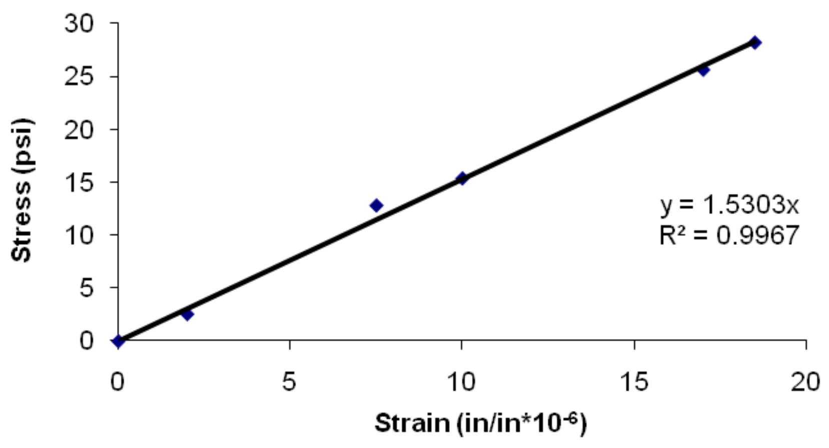


Figure 33. Stress - strain relationship for catapult arm under axial loading.

Data From the CompactRIO

The LabView program Virtual Instrument (VI) was structured in a producer-consumer loop as shown in Figure 34. The purpose of this structure was to take readings at a very fast sampling rate while writing the data to a text file. We had initially attempted to use just a timed loop structure, however writing to the data file and displaying the measurement on the front panel sometimes interfered with the scan rate. A producer-consumer loop has all time-critical data occurring in the producer loop, which queues up data in memory for the consumer loop. The consumer loop then executes when adequate processing power is present – which doesn't interfere with the sampling rate. In our case, the data sampling occurred in the producer loop, while the data writing and waveform display occurred in the consumer loop.

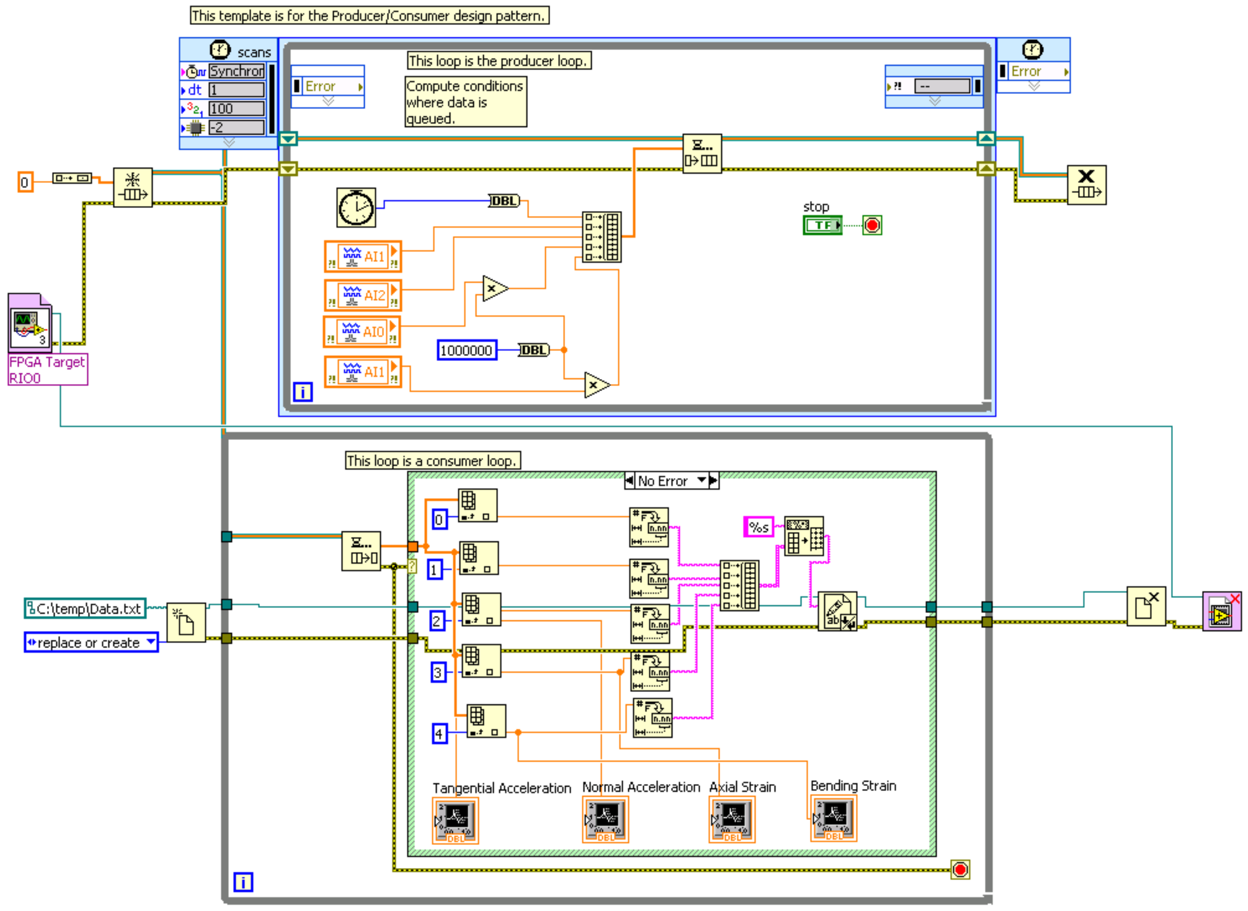


Figure 34. LabView VI for obtaining signals in producer-consumer format.

MatLab, Post-processing, and Results

After the data were recorded using this producer-consumer structure, a post-processing MatLab code was used to interpret the results. Shown in Figure 35, the actual catapult motion time was only a very small portion of the entire sampling time. This was because each trial was initiated by pressing run, releasing the catapult, and allowing adequate time for the program to write the data. However, we needed a way to easily identify the duration of the catapult motion.

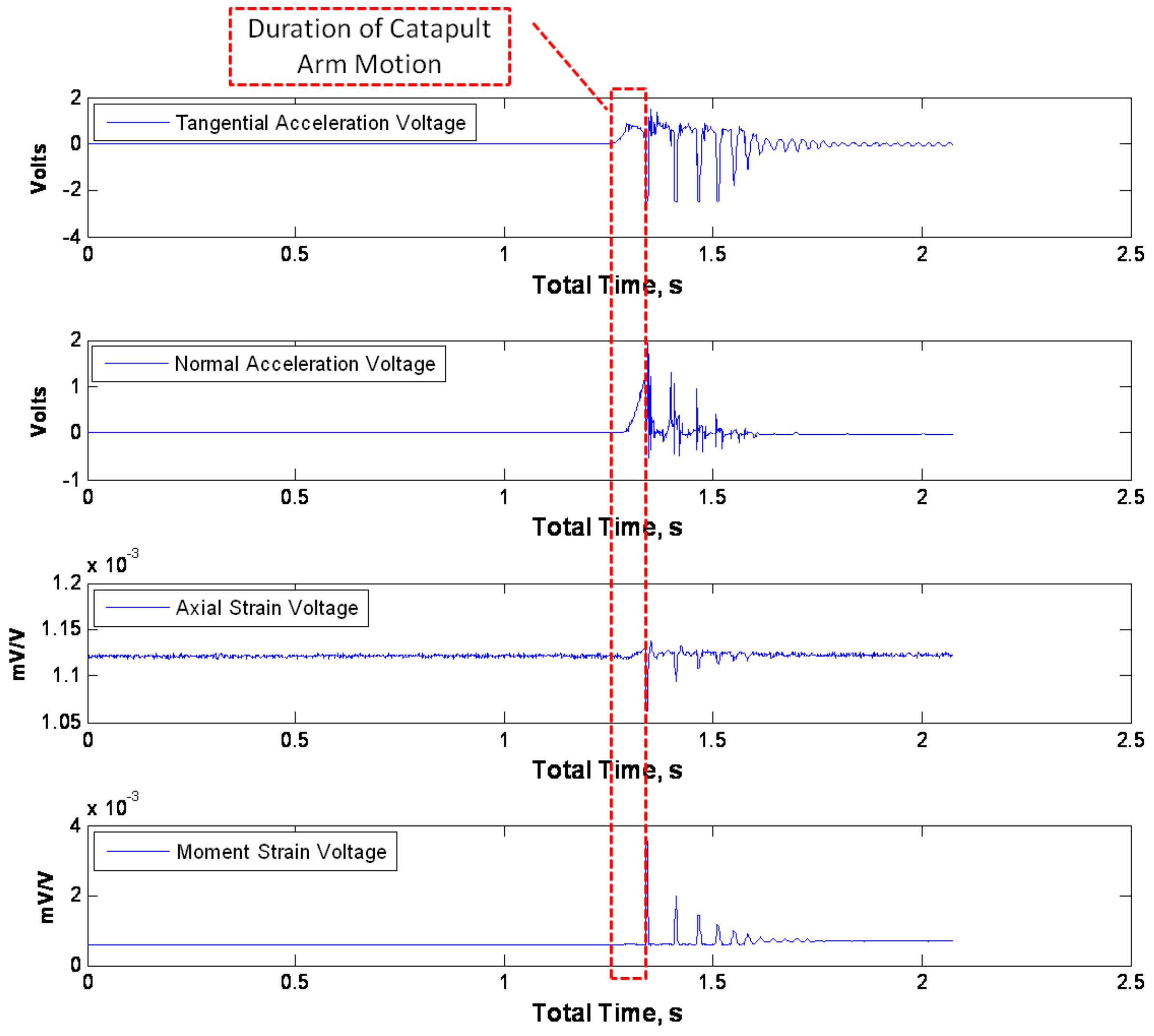


Figure 35. MatLab output for raw voltages obtained from LabView.

We accomplished this by programming the MatLab code attached in Appendix I , which searched for trigger values of normal and tangential acceleration. When both readings for acceleration surpassed their trigger values, the beginning of catapult motion was indicated. When the trigger value for tangential acceleration became negative, this indicated the end of the catapult motion – hitting the stopper pin. The first plot of Figure 36 illustrates the entire recording of catapult acceleration data, from the time the time the LabView recording is started to when it is stopped. This data include the period of no movement, the catapult motion, and the oscillations after the arm has hit the stopper pin. However in this case, the catapult motion itself occurs between approximately 1.25 and 1.3 seconds. Using the aforementioned trigger values, the second two plots of Figure 36 show a rescaled time, which highlights the catapult motion by itself, and rescales the time to zero. The experimental angular velocity and angular acceleration were calculated from the experimental tangential and normal accelerations using Equations (3) and (4).

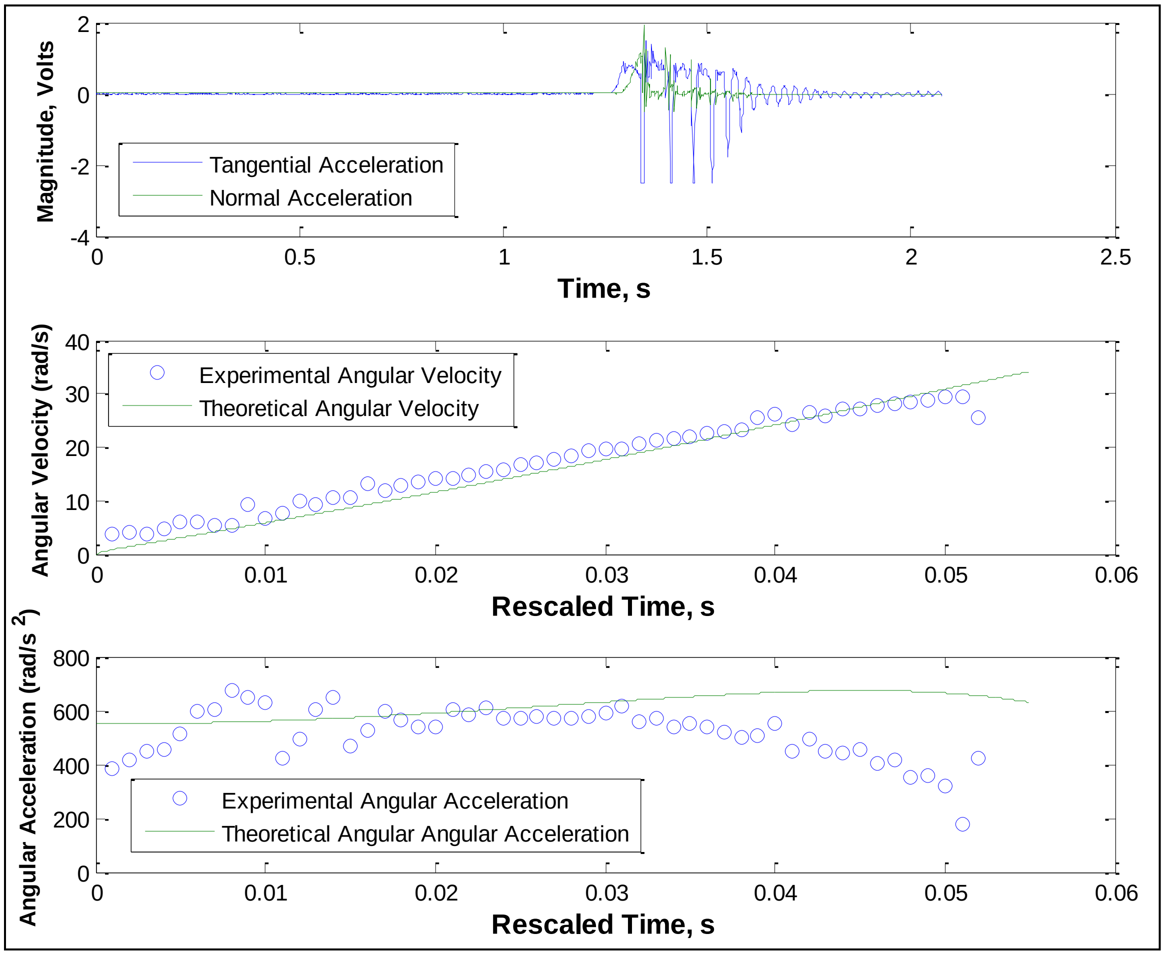


Figure 36. MatLab output of tangential acceleration, normal acceleration, angular velocity, and angular acceleration with corresponding theoretical results for a catapult pullback angle of 180° and stopper angle of 125°.

As seen in the figure, the experimental angular velocity matches very closely to the theoretical angular velocity obtained using the code in Appendix J . However, both the maximum angular velocity and acceleration are somewhat overestimated by the theoretical model. I believe that this is because during the final part of the catapult motion, a part of the rubber band remains in its “stretched” state due to the friction from the rubber band pin, shown in Figure 37. The theoretical model, in contrast, assumes that the entire rubber band unstretches evenly. After analyzing some high-speed catapult footage, it appeared that a section of the rubber band did in fact remain stretched during the catapult motion. Because of this, it is possible that not all the potential energy stored in the entire rubber band length is converted to kinetic angular velocity.

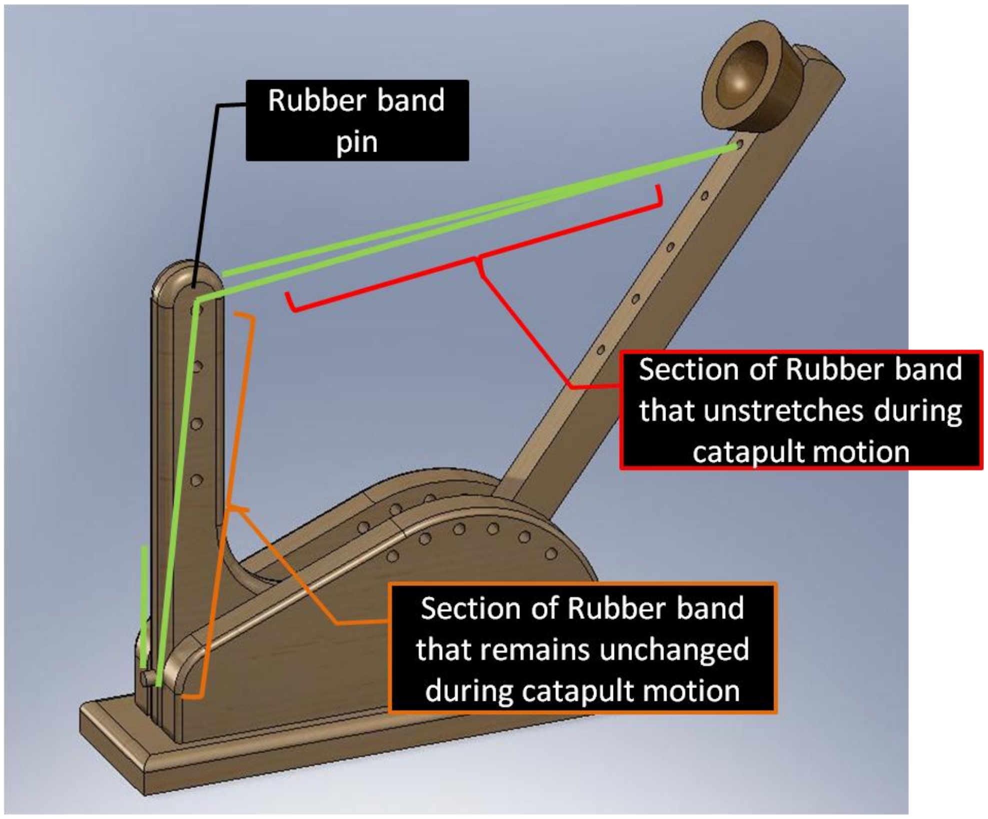

Rubber band pin
Section of Rubber band that unstretches during catapult motion
Section of Rubber band that remains unchanged during catapult motion

Figure 37. Diagram of unstretching and unchanged portions of rubber band during catapult motion.

After analyzing some initial strain gage results, we realized that the strain measurement for a no-load condition changed every time. There appeared to be an offset after every trial; therefore a study was conducted to see if any residual strain was present after each catapult launch. Shown in Figure 38 is the axial strain study for the catapult arm, using the P3 Strain Indicator and Recorder. We can see a linear strain increase of about 0.5 $\mu\varepsilon$ per trial. Ideally, we would want to re-zero the catapult every time to the no-load state. However, since many students could potentially be taking the data during launch day, we needed a more efficient way of re-zeroing the strains. This could potentially be accomplished by simply matching the initial strain magnitudes of the theoretical and experimental strains.

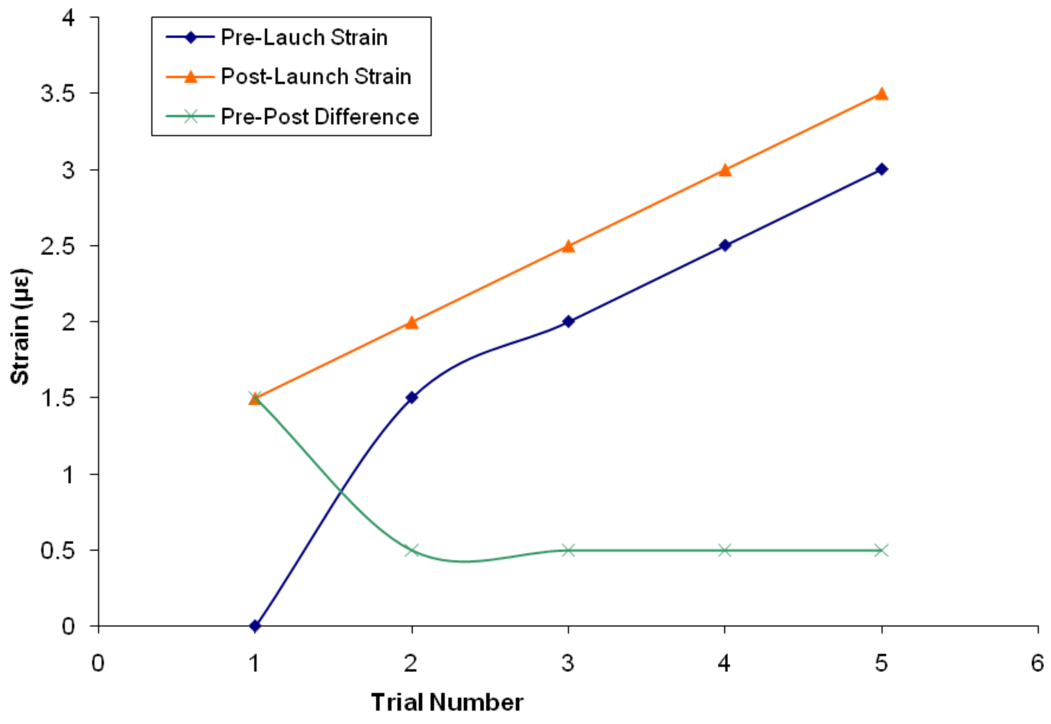


Figure 38. Residual axial strain study for catapult arm for five catapult launches.

Shown in Figure 39 is the MatLab output for the results of the axial and moment strains for a single trial. The first plot in the figure shows the raw voltage in $mV_{output}/V_{excitation}$ for each of the Wheatstone bridges. Shown in the second two plots are the axial strain and moment strain, rescaled as mentioned previously, and calculated using the below equation (National Instruments 2009).

$$\frac{V_o}{V_{excitation}} = \frac{S_g \epsilon}{2} \tag{7}$$

From the first two plots of Figure 39, it can be seen that strain gages did pick up the impact and the resultant oscillations afterwards. However, as shown in the third and fourth plots, it appeared that the actual axial and moment strain of the catapult arm was much smaller than the theoretical model. Also, a lot of noise was present in the signal, due to the measured strain being so small. Because of this, not very much useful strain data could be obtained from the duration of free catapult motion.

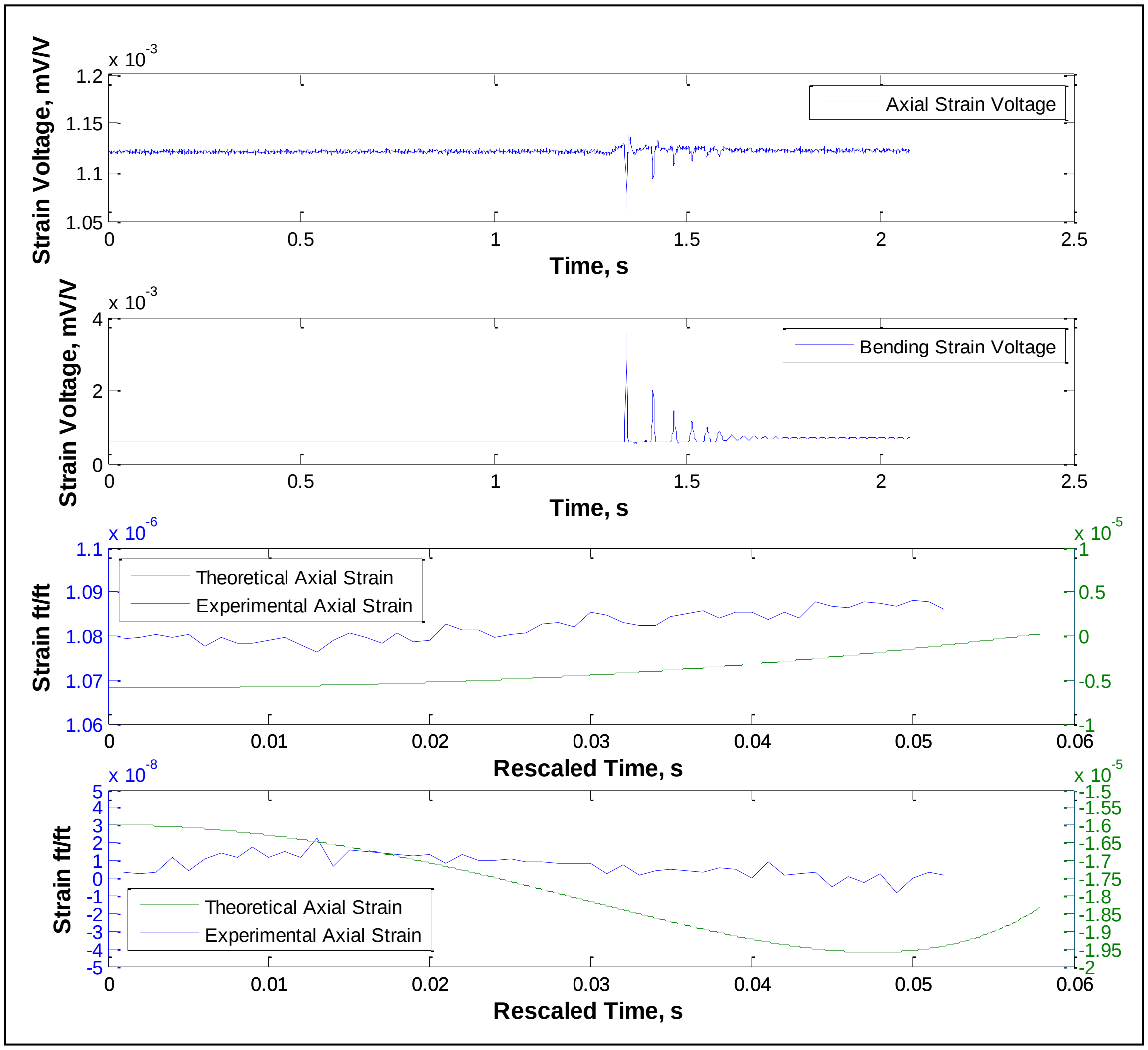


Figure 39. MatLab output of axial strain and moment strain with corresponding theoretical results for a catapult pullback angle of 180° and stopper angle of 125°.

Even so, the strain *profiles* in the third and fourth plots of Figure 39 show a resemblance with the theoretical model, although they are both noisy and off by orders of magnitude. This means that the strain could have been be performing as expected, however the strain gage was not sensitive enough to make a precise measurement. The strains however, despite having an unexpected magnitude, were used to determine useful information about the impact time. Shown in Figure 40 is the estimated duration of impact, using the moment strain output. The start of the plot was obtained by finding the time where the angular acceleration became negative, and ended when the moment strain returned to its pre-impact state.

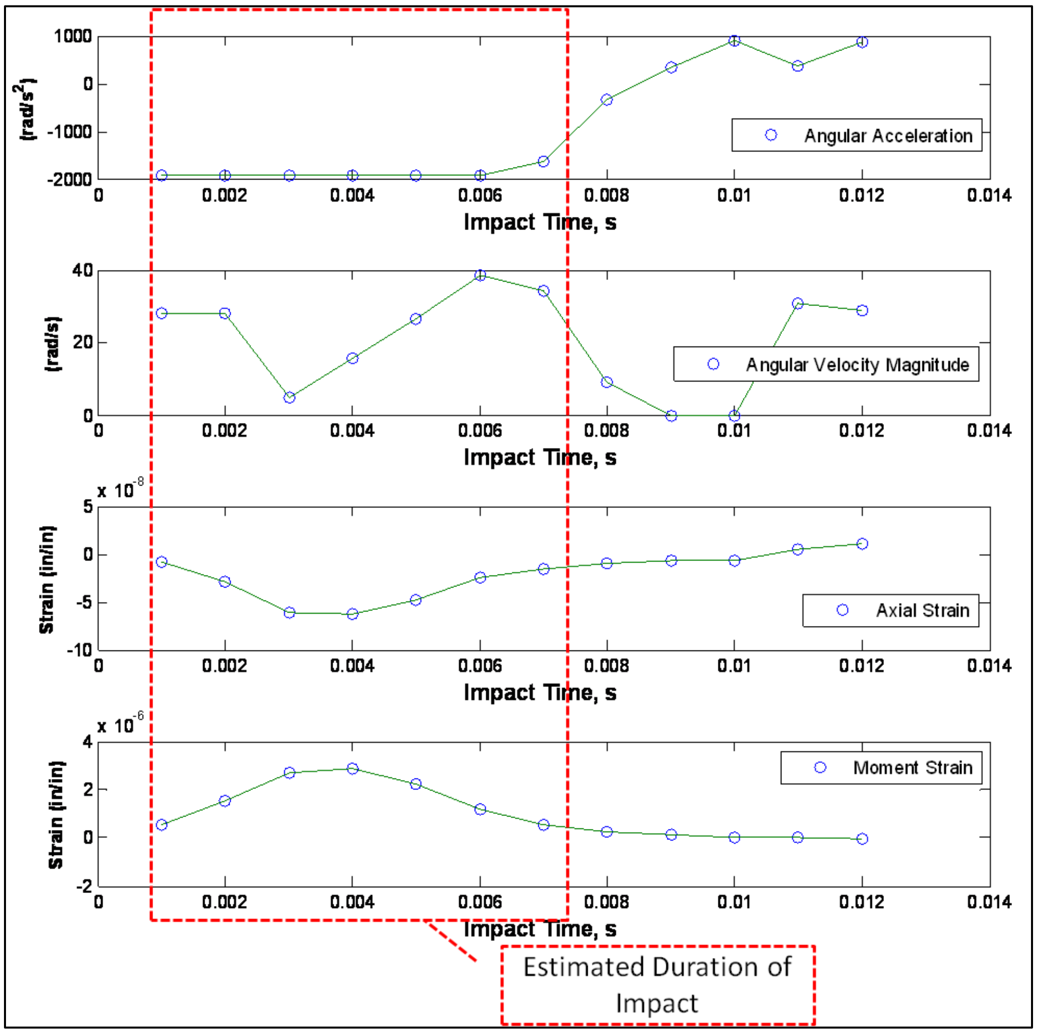


Figure 40. MatLab output of axial strain, moment strain, angular acceleration, and angular velocity magnitude during the impact, with catapult pullback angle of 180° and stopper angle of 125°.

The experimental results for angular acceleration were much lower than expected during the impact, shown in the first plot of Figure 40. Theoretically, integrating the angular acceleration from zero to the time it takes for the catapult arm to approach a zero velocity should equal the angular velocity just before the impact. However, since our ADXL-278 is only rated for ±50g, our angular acceleration is actually outside the maximum range that can be measured by the accelerometer. For example, a linear change from ω=35rad/s to 0rad/s in 0.003 seconds would correspond to an constant angular acceleration of 3920rad/s^2. Based on the positioning of the accelerometer, this angular acceleration corresponds to a linear tangential acceleration of nearly 400g's. Also, we see that the final angular velocity magnitude is higher than the initial angular velocity magnitude during the impact, which is impossible. This was likely due to the extreme spike in acceleration, which caused the accelerometer to operate outside of its intended rated range. The vibration that occurred during the impact also could have caused incorrect readings.

Because of these incorrect readings of angular acceleration and angular velocity, the time of impact determined from the moment strain profile was used to roughly estimate the force on the stopper pin. Ideally, we would have used the change in the experimentally measured velocity with respect to time. However as we mentioned previously, these values were ultimately incorrect. Instead, using impulse momentum principles, we estimated the impact force by simplifying equation (8) to equation (9). This simplification was done by assuming that the force was constant during impact and that the collision was perfectly elastic, setting $\omega_{impact\_final} = - \omega_{impact\_initial}$. The case shown in Figure 30 would correspond with an average stopper pin force of approximately 250 lbs.

$$\int_{t_1}^{t_2} \sum M_O \, dt = \Delta H \tag{8}$$

$$F_{avg} = \frac{2I\omega_{impact\ \_initial}}{t * L_{stoppe\ r}} \tag{9}$$

Catapult Instrument Summary and Recommendations

Overall, we accomplished the goals of the instrumentation that we initially set out to complete. Although some equipment, such as the strain gages, did not function as we intended, all of the instruments provided meaningful data that were used to quantify the physical catapult. However, there are several recommendations that can be made after my experience with this first attempt. First, another data acquisition system should be used. The task of acquiring the signals could have been more easily done with a CompactDAQ, which is both cheaper and can scan more quickly in real-time mode. An increased scan rate would result in a better position calculation, and could possibly better capture the data during impact time. Second, the strain gages should be set up to measure a larger magnitude of strain, by either hallowing out some material to increase stress and strain, or by using a different strain gage. This would allow for a much better signal-to-noise ratio, which was very high during our trials. In addition to increasing the strain magnitude, some filtering technique could also be investigated. However at the moment, the strain magnitude is too small for any kind of filtering.

We also attempted to compensate for the effect of the nonuniform unstretching of the rubber band by modeling a percentage of energy that was dissipated, as well as changing the theoretical stretch distance. Both methods had a worse result than the original model. For future runs, the rubber band should be pinned at the point of rotation and a force-displacement curve measured.

The current LabView Virtual Instrument could also be set up for an external trigger if desired. This way of acquiring the data could have LabView constantly taking readings, with an external trigger signaling to write the next few seconds of measurements to a file. This method could become useful if the current system is too slow for students to use during launch day.

The instrumentation should be effective in providing students for an additional means of self-assessment. By furthering the principles of self-assessment and reality, it has the potential to make the catapult project a better and more effective MEA. In addition, we now know the time of impact of the catapult arm, which was previously just assumed.

Overall Response to MEAs

The student responses to VAR and Catapult MEAs were assessed by a post-course evaluation survey of 23 questions. The questions asked in Figure 41 and Figure 42 simply asked if a student agreed with the statement shown. As shown in Figure 41, the overall response was positive, with most students agreeing that the MEAs were motivational learning tools. However, when we examined the responses to traditional assignments, students felt that the individual homework assignments helped them learn the material better, shown in Figure 42. We believe that this was because some students were accustomed to a more traditional type of learning, which focused on individual textbook problem solving. Figure 42 also shows some student resistance to team-type assignments, which could have influenced their opinion on MEAs.

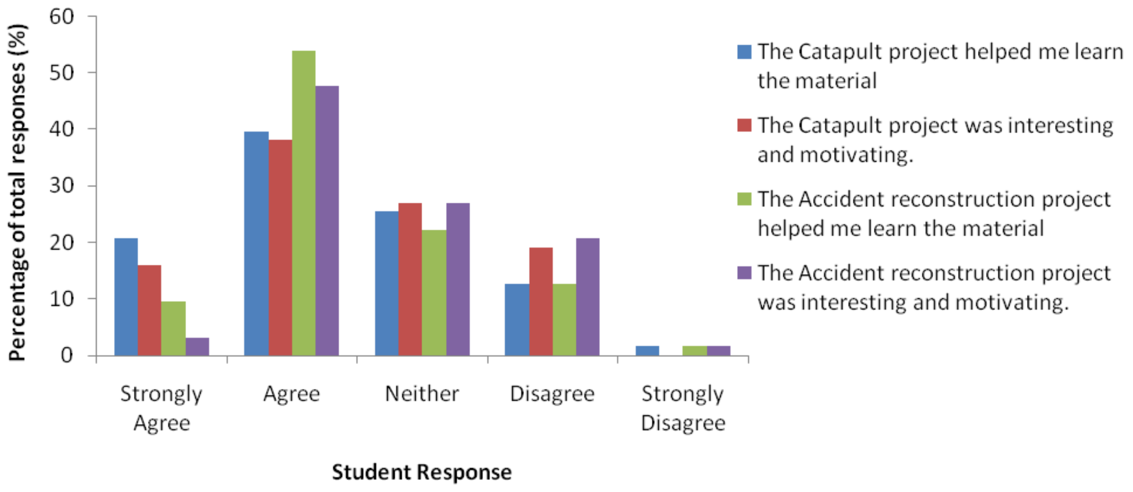


Figure 41. Student responses for the MEA projects for Spring quarter.

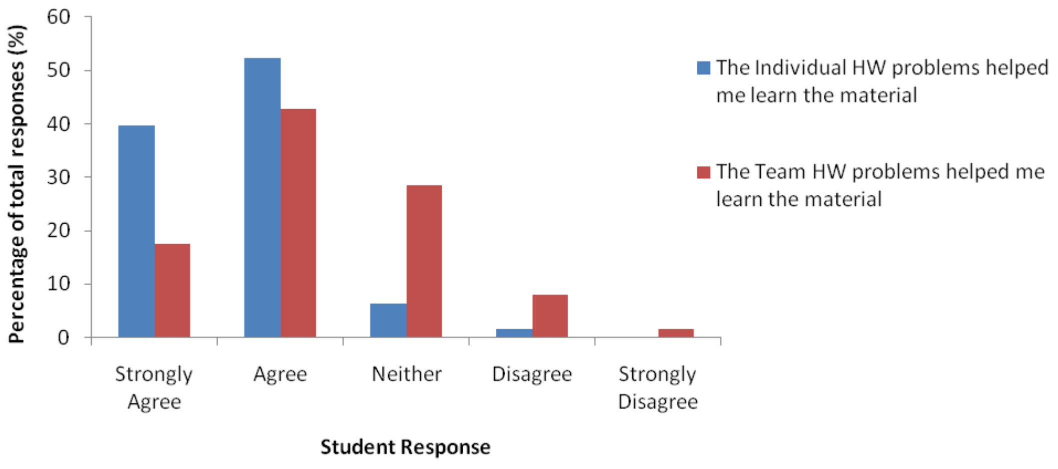


Figure 42. Student responses for HW assignments for Spring quarter.

The post-course survey question "*Some students seemed to lack motivation for the class. We tried to do the real world projects and show different applications of the material. What else should we do to increase student motivation?*" required a written-response, and generated some very interesting responses. There were three typical student responses to this question. The first was very positive, indicating that the student had benefitted greatly from the projects, and was grateful for the class experience. Several students recognized that those projects required heavy instructor time commitment, and explicitly thanked the professor. The second response was neutral, saying that the projects neither helped nor hindered their learning. Some stated that nothing could motivate them because they were not interested in subject matter at all. The third student response was negative, saying that the project was "too much work" and irrelevant to what he was learning.

Focusing specifically on the VAR MEA, a thematic analysis was conducted for two survey questions. These questions asked for written responses from 258 students in two quarters of Dynamics courses. The first question asked, "What did you like about the [VAR] Project and why?" Shown in Figure 43 are the comments sorted into six major categories. Fifty percent of the responses indicated that students enjoyed having a realistic context. Seventeen percent stated specifically that students liked either the case report format, the client setting, or the overall assignment structure. Another seventeen percent of the comments stated that the project helped students learn the principles of work-energy and momentum. The final fifteen percent of the comments stated that students enjoyed the group aspect of the project.

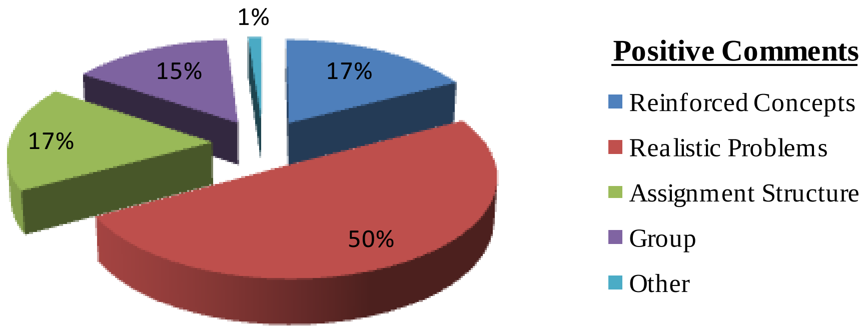


Figure 43. Responses to the survey question “What did you like about the [VAR] Project and why?”

The next question asked, “What didn’t you like about the [VAR] Project and why?” These comments were also broken down into six major categories, as shown in Figure 44. Thirty-four percent of the comments were critical of the overall vagueness of the problem statements and the information provided. Twenty-five percent of the comments were complaints of the heavy time commitment or the difficulty of the project. Twenty percent were complaints of the team aspect, with students indicating that they had difficulty working with their groups. The remaining six and seven percent of the comments were critical of the grading criteria and the increased writing efforts of the memo, respectively.

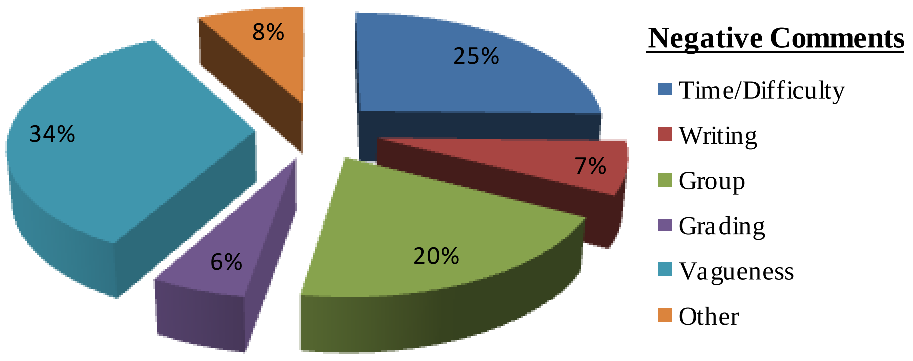


Figure 44. Responses to the survey question "What didn't you like about the [VAR] Project and why?"

Conclusion

The development of the Catapult and VAR MEAs have been a significant task for the research team over the course of the past year. Over the past three quarters, we have been able to refine the projects, making them better teaching tools, and better MEAs. The VAR GUI was intended to reduce instructor time commitment, while facilitating the use of the VAR MEA for other universities. The Catapult instrumentation was intended to expand upon the reality and self-assessment principles of the MEA by allowing students to connect their physical project with experimental data. While we initially thought that VAR GUI and the Catapult instrumentation would be relatively simple, once we began, we quickly realized that they would require some serious time commitment.

Students who were exposed to the VAR MEA scored noticeably higher in the Dynamics Concept Inventory exam than students who were not. Because of this, we were encouraged to develop the VAR GUI to help cut down the instructor and teaching assistant development time. Currently, since we are operating on the CCLI grant, we are able to dedicate several people to developing and assessing the MEA. However, we wanted this project to carry on long after our grant was finished. When only a single instructor is responsible for developing new VAR cases and grading them, the GUI will greatly reduce the amount of time he will have to commit. The GUI can help develop new case scenarios, and check the validity of student models.

The Catapult instrumentation was, by far, one of the most time consuming developments in the MEA program. We were glad to see that we could capture some valid data for the angular acceleration and velocity, despite the strain gages not functioning as we intended. Even so, we believe that it will provide for great self-assessment tool for students to check their theoretical models. Some of the development time could have been reduced by acquiring some new equipment. For example, using a simple USB-based multi-channel data acquisition system would have eliminated programming of a relatively complex LabView instrument. However to keep costs down, we utilized what was available. Since the catapult instrumentation has yet to be tested in a classroom, no student responses are available for the instrumentation itself. However, we are confident that students will appreciate being able to validate their own theoretical model with the actual data of their launch. All of the hardware and software has been designed to operate very simply. Ideally, each student team will be able to take an individual set of data for each launch. Using their experimental data, they will be able to validate and refine their model accordingly.

We believe that the MEAs that we have implemented have improved the student experience at Cal Poly. As stated previously, the overall student response to the MEAs have been positive. However, when students are exposed to a different kind of teaching than they are traditionally accustomed to, some resistance to the change is to be expected. Even so, we hope that more students will learn to accept that these MEAs really do have a positive impact on their learning, and better prepare them to their future professions in industry.

Works Cited

Boyer Commission on Education Undergraduates in the Research University. *Reinventing Undergraduate Education: A Blueprint for America's Research Universities.* Stony Brook, NY: State University of New York at Stony Brook for the Carnegie Foundation for the Advancement of Teaching, 1998.

Chamberlin, Scott A. "How does the Problem-Based Learning Approach Compare to the Model-Eliciting Activity Approach in Mathematics Instruction?" *International Journal of Mathematics Teaching and Learning*, 2008.

Goodwin, Lora. "Is There a Correlation Between Conceptual Understanding and Procedural Knowledge in Introductory Dynamics?" *ASEE PSW Conference.* San Diego, CA: ASEE, 2009.

Gray, Gary L. "The Dynamics Concept Inventory Assessment Test: A Progress Report and Some Results." *American Society for Engineering Education Annual Conference & Exposition.* Portland, Oregon: American Society for Engineering Education, 2005. 1.

J.L. Meriam, L.G. Kraige. *Engineering Mechanics Dynamics.* New York, NY: John Wiley & Sons, Inc., 2002.

Jolley, William O. "A Fun and Challenging Engineering Dynamics Project Using a Lego Construction Set." *American Society for Engineering Education Annual.* Nashville, Tennessee: American Society for Engineering Education, 2003. 1.

Lesh, Richard. *Beyond Constructivism: Models and Modeling Perspectives on Mathematics.* Mahwah, NJ: Lawrence Erlbaum Associates, 2004.

—. *Handbook of Research Design in Mathematics and Science Education.* Mahwah, NJ: Lawrence Erlbaum Associates, 2000.

National Instruments. *Measuring Strain with Strain Gages.* June 26, 2009. http://zone.ni.com/devzone/cda/tut/p/id/3642 (accessed August 18, 2009).

—. *Using CompactRIO Scan Mode with Unsupported Backplanes.* November 13, 2008. http://digital.ni.com/public.nsf/allkb/122E971F52FD081A86257500007A046C (accessed August 16, 2009).

Prince, Michael. "Does Active Learning Work? A Review of the Research." *Journal of Engineering Education*, 2004.

Reffeor, Wendy. "Incorporating Design in an Introduction to Dynamics Course." *ASEE Annual Conference and Exposition.* Montréal, Quebec, Canada: ASEE, 2002.

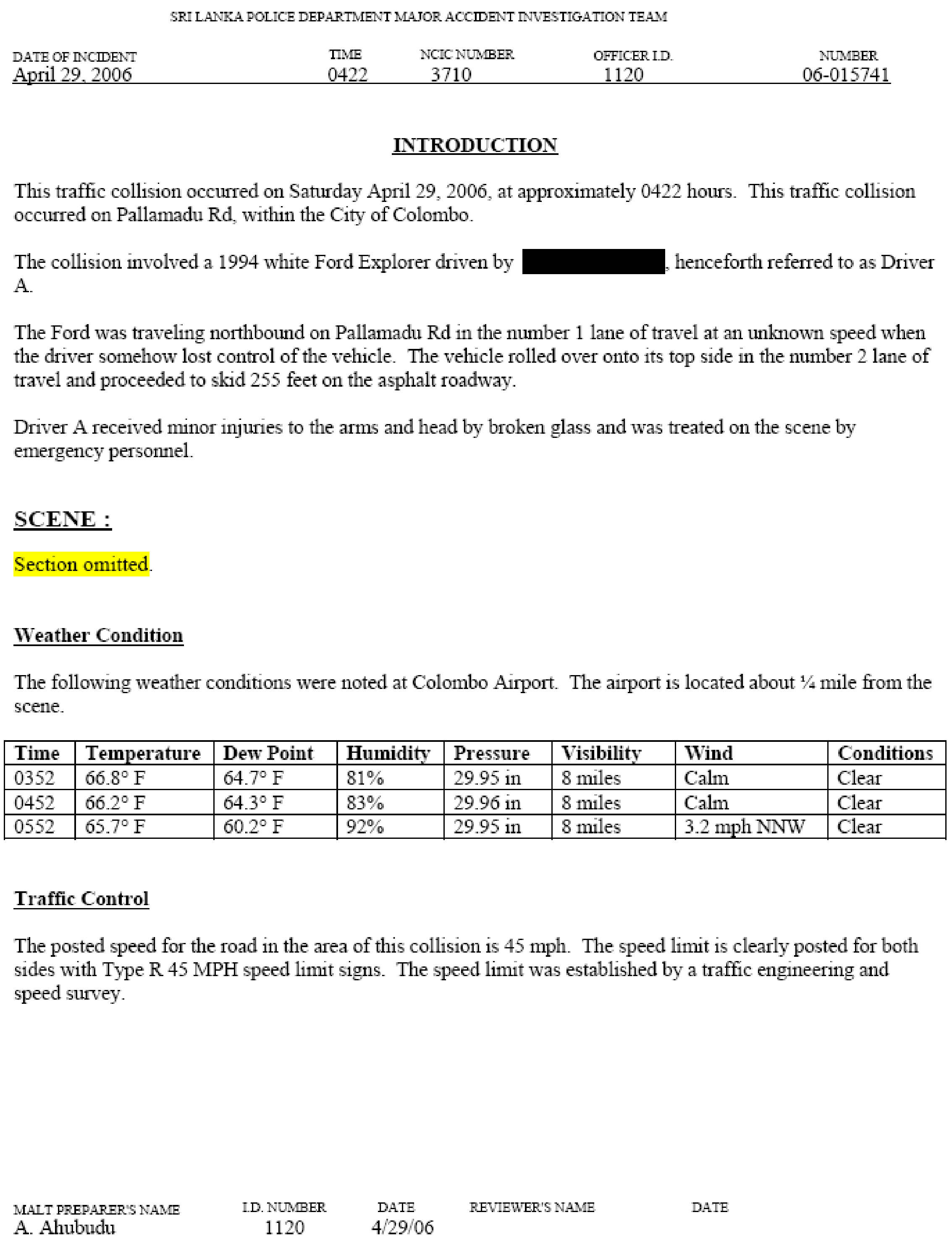
Savery, John R. "Overview of Problem-based Learning: Definitions and Distinctions." *Interdisciplinary Journal of Problem-based Learning: Vol. 1: Issue 1, Article 3,* 2006.

Self, Brian. *National Science Foundation: Collaborative Research: Improving Engineering Students' Learning Strategies through Models and Modeling.* August 31, 2007. http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0717595 (accessed August 18, 2009).

Smithsonian Institution. *Smithsonian Physical Tables.* Washington, D.C.: Smithsonian Institution Press, 1969.

Vishay Micro-Measurements. *Instruction Bulletin B-137-16: Strain Gage Applications with M-Bond AE-10, AE-15, and GA-2 Adhesive Systems.* Don Mills, ON, February 4, 2005.

Appendix A VAR Case 1



SRI LANKA POLICE DEPARTMENT MAJOR ACCIDENT INVESTIGATION TEAM

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
April 29, 2006	0422	3710	1120	06-015741

INTRODUCTION

This traffic collision occurred on Saturday April 29, 2006, at approximately 0422 hours. This traffic collision occurred on Pallamadu Rd, within the City of Colombo.

The collision involved a 1994 white Ford Explorer driven by [redacted], henceforth referred to as Driver A.

The Ford was traveling northbound on Pallamadu Rd in the number 1 lane of travel at an unknown speed when the driver somehow lost control of the vehicle. The vehicle rolled over onto its top side in the number 2 lane of travel and proceeded to skid 255 feet on the asphalt roadway.

Driver A received minor injuries to the arms and head by broken glass and was treated on the scene by emergency personnel.

SCENE :

Section omitted.

Weather Condition

The following weather conditions were noted at Colombo Airport. The airport is located about ¼ mile from the scene.

Time	Temperature	Dew Point	Humidity	Pressure	Visibility	Wind	Conditions
0352	66.8° F	64.7° F	81%	29.95 in	8 miles	Calm	Clear
0452	66.2° F	64.3° F	83%	29.96 in	8 miles	Calm	Clear
0552	65.7° F	60.2° F	92%	29.95 in	8 miles	3.2 mph NNW	Clear

Traffic Control

The posted speed for the road in the area of this collision is 45 mph. The speed limit is clearly posted for both sides with Type R 45 MPH speed limit signs. The speed limit was established by a traffic engineering and speed survey.

MALT PREPARER'S NAME	I.D. NUMBER	DATE	REVIEWER'S NAME	DATE
A. Ahubudu	1120	4/29/06		

SRI LANKA POLICE DEPARTMENT MAJOR ACCIDENT INVESTIGATION TEAM

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
April 29, 2006	0422	3710	1120	06-015741

VEHICLES

Vehicle One (1994 Ford Explorer)

Description

Year:	1994
Make:	Ford
Model:	Explorer
License:	[redacted]
VIN:	[redacted]
Engine:	1.5L V4
Transmission:	5 speed Manual
Color:	White
Type:	2 door
Weight:	4580 pounds
Length:	174.5 inches (4673 mm)
Height:	67.5 inches (1714 mm)
Width:	70.2 inches (1778 mm)
Center of gravity:	24.1 inches (height)

Damage Description:

Front:

Minor to moderate damage was sustained to the front right portion of V1.

Right

Minor to moderate damage was sustained to this portion of V1. This damage consisted of scrapings where V1 was in contact with the road and broken side windows.

Left

I did not observed any damage to this portion of V1.

Rear

I did not observed any damage to this portion of V1.

Roof

Moderate damage was sustained to the roof of V1 but the average height of V1 remained unchanged.

MALT PREPARER'S NAME	I.D. NUMBER	DATE	REVIEWER'S NAME	DATE
A. Ahubudu	1120	4/29/06		

Appendix B VAR Case 2

SRI LANKA POLICE DEPARTMENT MAJOR ACCIDENT INVESTIGATION TEAM

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
June 20, 2006	0518	3710	1120	06-017742

INTRODUCTION

This traffic collision occurred on Friday June 20, 2006, at approximately 0518 hours. This traffic collision occurred on Jawatte Rd, within the City of Colombo.

The collision involved a 1999 red Nissan Super Saloon driven by [redacted], and a 1994 black Ford Fiesta driven by [redacted].

The Nissan was traveling northbound on Jawatte Rd up a 7% grade. As the Nissan reached the top of the grade it collided head on with the Ford which was traveling southbound. The road north of the collision point, on which the Ford had been traveling, had a 0% grade. After impact, both vehicles slid together with locked wheels 5.8 meters down the hill.

Prior to the accident, a third driver reported that the Nissan was travelling approximately 18 - 25 km/h. The officer then left to respond to another incident on Kelaniya Rd.

Physical evidence at the scene indicated that the driver of the Ford was aware that he was about to impact Nissan. Wheel locked skid marks just prior to the collision were measured to be 9.4 meters in length and matched the tire pattern of the Ford. Roadway conditions at the time of the accident were slightly wet.

SCENE :

Section omitted.

Weather Condition

The following weather conditions were noted at Colombo Airport. The airport is located about 3.2 km from the scene.

Time	Temperature	Dew Point	Humidity	Pressure	Visibility	Wind	Conditions
0452	19.3° C	18.2° C	81%	760.7 mm H_20	12.9 km	Calm	Light Rain
0552	16.8° C	17.9° C	83%	761.0 mm H_20	12.9 km	Calm	Foggy
0652	18.7° C	15.7° C	92%	760.7 mm H_20	12.9 km	5.1 km/h NNW	Foggy

Traffic Control

The posted speed for the road in the area of this collision is 40 km/h. The speed limit is clearly posted for both sides with Type R 40 KM/H speed limit signs. The speed limit was established by a traffic engineering and speed survey.

VEHICLES

MALT PREPARER'S NAME	I.D. NUMBER	DATE	REVIEWER'S NAME	DATE
A. Ahubudu	1120	6/20/06		

SRI LANKA POLICE DEPARTMENT MAJOR ACCIDENT INVESTIGATION TEAM

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
June 20, 2006	0518	3710	1120	06-017742

Vehicle One (1999 Nissan Super Saloon)

Description

Year:	1999
Make:	Nissan
Model:	Super Saloon
License:	[redacted]
VIN:	[redacted]
Engine:	2200cc V4
Transmission:	5 speed Manual
Color:	Red
Type:	4 door
Weight:	1225 kg

Damage Description:

Front:

There was moderate to severe damage to this portion of V 1. There was crumpling and creasing to the hood and sub-frame, along with breaks to the plastic front grill. Both headlights were broken out. An examination of the broken bulbs showed oxidation and melting to the filament. This indicated that the headlights were in the "On" position at the time of this collision. Some of the engine fluids (oil, coolant, brake fluid, etc.) had spilled onto the roadway at the collision scene.

Right

V1 sustained no visible damage to this end.

Left

V1 sustained no visible damage to this end.

Rear

V1 sustained no visible damage to this end.

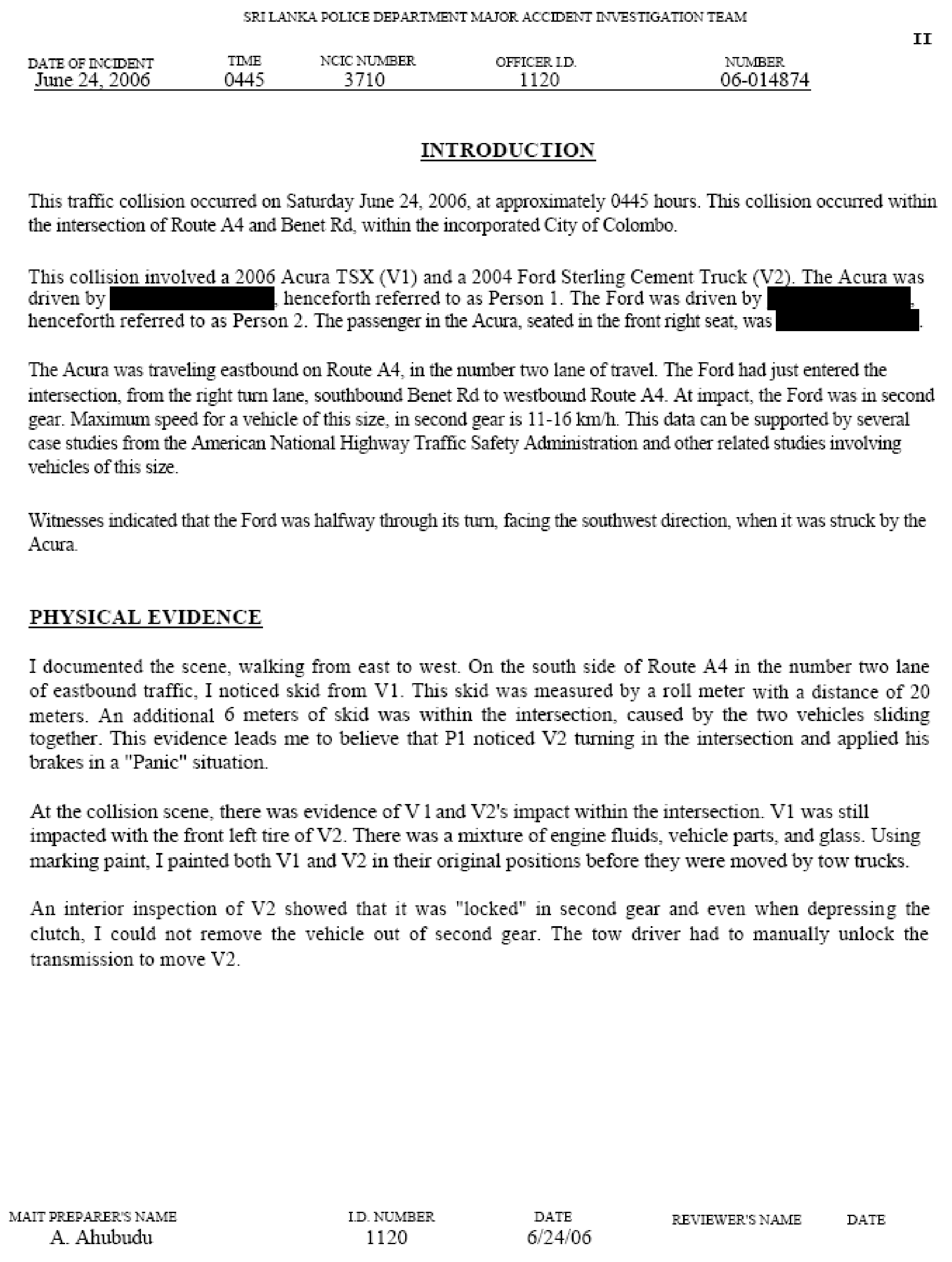
Roof

V1 sustained no visible damage to this end.

VEHICLES (Continued)

MALT PREPARER'S NAME	I.D. NUMBER	DATE	REVIEWER'S NAME	DATE
A. Ahubudu	1120	6/20/06		

Appendix C VAR Case 3



SRI LANKA POLICE DEPARTMENT MAJOR ACCIDENT INVESTIGATION TEAM

II

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
June 24, 2006	0445	3710	1120	06-014874

INTRODUCTION

This traffic collision occurred on Saturday June 24, 2006, at approximately 0445 hours. This collision occurred within the intersection of Route A4 and Benet Rd, within the incorporated City of Colombo.

This collision involved a 2006 Acura TSX (V1) and a 2004 Ford Sterling Cement Truck (V2). The Acura was driven by [redacted], henceforth referred to as Person 1. The Ford was driven by [redacted], henceforth referred to as Person 2. The passenger in the Acura, seated in the front right seat, was [redacted].

The Acura was traveling eastbound on Route A4, in the number two lane of travel. The Ford had just entered the intersection, from the right turn lane, southbound Benet Rd to westbound Route A4. At impact, the Ford was in second gear. Maximum speed for a vehicle of this size, in second gear is 11-16 km/h. This data can be supported by several case studies from the American National Highway Traffic Safety Administration and other related studies involving vehicles of this size.

Witnesses indicated that the Ford was halfway through its turn, facing the southwest direction, when it was struck by the Acura.

PHYSICAL EVIDENCE

I documented the scene, walking from east to west. On the south side of Route A4 in the number two lane of eastbound traffic, I noticed skid from V1. This skid was measured by a roll meter with a distance of 20 meters. An additional 6 meters of skid was within the intersection, caused by the two vehicles sliding together. This evidence leads me to believe that P1 noticed V2 turning in the intersection and applied his brakes in a "Panic" situation.

At the collision scene, there was evidence of V1 and V2's impact within the intersection. V1 was still impacted with the front left tire of V2. There was a mixture of engine fluids, vehicle parts, and glass. Using marking paint, I painted both V1 and V2 in their original positions before they were moved by tow trucks.

An interior inspection of V2 showed that it was "locked" in second gear and even when depressing the clutch, I could not remove the vehicle out of second gear. The tow driver had to manually unlock the transmission to move V2.

MAIT PREPARER'S NAME	I.D. NUMBER	DATE	REVIEWER'S NAME	DATE
A. Ahubudu	1120	6/24/06		

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
June 24, 2006	0445	3710	1120	06-014874

VEHICLES (Continued)

Damage:

Front:

The damage sustained to this portion of V 1 consisted of the entire front bumper being removed from the vehicle. The right side headlight assembly was completely broken out. An inspection of the broken headlight assembly, with the exposed headlight bulb filament, showed signs of oxidation and melting of the filament. This evidence showed that P1 had the headlights of V1 in the "On" position. The hood and right quarter panel was crumpled and dented. V1 had also expelled some of its engine fluids, i.e. Radiator fluid, oil, brake fluid.

Right:

The only damage sustained on this side of V 1 was the front right quarter panel. This damage consisted of the quarter panel being dented and crushed.

Left:

Besides slight crumpling and warping to the front left quarter panel, no other damage was sustained to this side of V1.

Rear:

I did not observe any damage to this portion of V1.

Roof:

I did not observe any damage to this portion of V1.

MAIT PREPARER'S NAME	I.D. NUMBER	DATE	REVIEWER'S NAME	DATE
A. Ahubudu	1120	6/24/06		

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
June 24, 2006	0445	3710	1120	06-014874

VEHICLES (Continued)

Vehicle 2 (V2, 2004 Ford Sterling)

Description:

Year:	2004
Make:	Ford
Model:	Sterling
License:	[redacted]
VIN:	[redacted]
Engine:	Mercedes MBE 4000 450
Transmission:	8LL
Color:	White
Vehicle Type:	3 Axle Short Pour Cement Mixer
Weight:	8436 kg

Damage:

Front:
No visible damage had occurred to this side.

Left:
No visible damage had occurred to this side.

Right:
Besides the front right wheel assembly being broken, with several air and fluid lines broken, no other visible damage occurred to this side.

Rear:
No visible damage had occurred to this side.

MAIT PREPARER'S NAME	I.D. NUMBER	DATE	REVIEWER'S NAME	DATE
A. Ahubudu	1120	6/24/06		

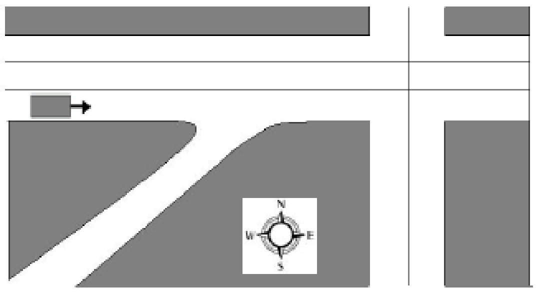
Appendix D VAR Case 4

SRI LANKA POLICE DEPARTMENT MAJOR ACCIDENT INVESTIGATION TEAM

12/26

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
July 21, 2006	0300	3710	1120	08-021828H

INTRODUCTION



On Thursday, July 21, at 0300 hours, a traffic collision occurred on the junction of B79 and B243 (both one way highways).

A 1996 Honda Accord (V1) driven by [redacted] and a 1998 Ford Focus (V2), driven by [redacted] were the two vehicles involved in this collision. These drivers will be henceforth referred to as Person 1 and Person 2, respectively.

V1 was traveling eastbound on B79 (one-way), when it struck V2.

Person 2 was transported to a hospital to be treated for major injuries. Person 1 was treated for minor injuries on-site.

Based on the statement of Person 1, he had been driving under the speed limit when he suddenly was struck by Vehicle 2, and claims that Vehicle 2 suddenly veered across the merging lane. He states he has no recollection after that. Person 1's speed was confirmed by a speed camera which captured his speed at 100 km/h. Person 2 claims he was slowly merging into eastbound traffic, when Person 1 slammed into him. No pre-collision data was recorded for Vehicle 2. The figure above shows the area in which the collision occurred.

PHYSICAL EVIDENCE

No skid marks were present for the vehicles before the impact zone. The post-collision skidmarks measured to be 80m for Vehicle 2 at an angle of 100° from North, in the Southeast direction. The post-collision skidmarks measured to be 100m for Vehicle 1, at an angle of 45° from north, in the Northeast direction. Both of these skid distances were measured from the impact zone to the final location of the vehicles.

The posted speed limit for both roads is 100km/h.

Roadway Description

Section Omitted

VEHICLES

Vehicle One (V1, 1995 Honda Accord)

Year:	1995
Make:	Honda
Model:	Accord EX
Color:	Green
License:	[redacted]
VIN:	[redacted]
Engine:	2700cc C27
Transmission:	4 speed automatic
Weight:	1295 kg

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
July 21, 2006	0300	3710	1120	08-021828H

Damage Description

Front:
V1 sustained heavy damage to the front driver's side.

Left:
Minor Dents and Scratches

Right:
Major damage was sustained on this side. Driver's-side door deflected at least a half-foot inwards.

Rear:
Various scrapes and dents.

Vehicle Two (V2, 1998 Ford Focus)

Year:	1998
Make:	Ford
Model:	Focus
Color:	Black
License:	[redacted]
VIN:	[redacted]
Engine:	1400cc Zetec-SE
Transmission:	5 Speed manual
Weight:	1364kg

Damage Description:

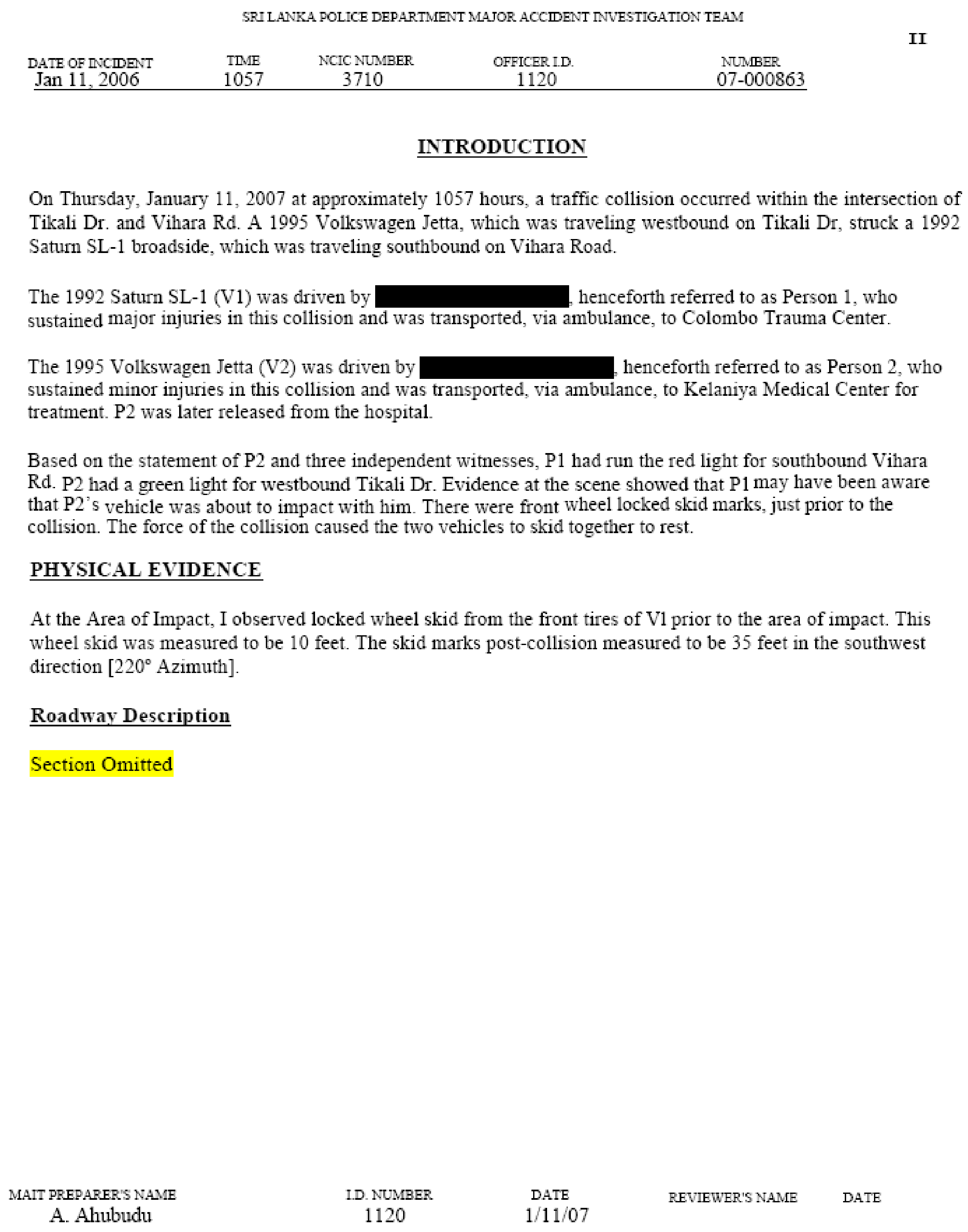
Front:
Heavy damage was sustained to the entire front fender of V2. Passenger side door crushed; passenger side fender missing.

Left:
Crushed appearance to all of left of vehicle – windows broken. Various scrapes and dents.

Right:
Various scrapes and dents.

Rear:
Various scrapes and dents.

Appendix E VAR Case 5



SRI LANKA POLICE DEPARTMENT MAJOR ACCIDENT INVESTIGATION TEAM

II

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
Jan 11, 2006	1057	3710	1120	07-000863

INTRODUCTION

On Thursday, January 11, 2007 at approximately 1057 hours, a traffic collision occurred within the intersection of Tikali Dr. and Vihara Rd. A 1995 Volkswagen Jetta, which was traveling westbound on Tikali Dr, struck a 1992 Saturn SL-1 broadside, which was traveling southbound on Vihara Road.

The 1992 Saturn SL-1 (V1) was driven by [redacted], henceforth referred to as Person 1, who sustained major injuries in this collision and was transported, via ambulance, to Colombo Trauma Center.

The 1995 Volkswagen Jetta (V2) was driven by [redacted], henceforth referred to as Person 2, who sustained minor injuries in this collision and was transported, via ambulance, to Kelaniya Medical Center for treatment. P2 was later released from the hospital.

Based on the statement of P2 and three independent witnesses, P1 had run the red light for southbound Vihara Rd. P2 had a green light for westbound Tikali Dr. Evidence at the scene showed that P1 may have been aware that P2's vehicle was about to impact with him. There were front wheel locked skid marks, just prior to the collision. The force of the collision caused the two vehicles to skid together to rest.

PHYSICAL EVIDENCE

At the Area of Impact, I observed locked wheel skid from the front tires of V1 prior to the area of impact. This wheel skid was measured to be 10 feet. The skid marks post-collision measured to be 35 feet in the southwest direction [220° Azimuth].

Roadway Description

Section Omitted

MAIT PREPARER'S NAME	I.D. NUMBER	DATE	REVIEWER'S NAME	DATE
A. Ahubudu	1120	1/11/07		

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
Jan 11, 2006	1057	3710	1120	07-000863

Weather Conditions

The following weather conditions were noted at the Colombo Airport.

Time	Temperature	Dew Point	Humidity	Pressure	Visibility	Wind	Conditions
10:29 AM	60.8° F/16.0° C	48.2° F/9.0° C	63%	29.96 in/ 1014.4 hPa	9.0 miles/ 14.5 km	8.1 mph/ 13.0 km/h South	Overcast
10:52 AM	61.0° F/16.1° C	48.0° F/8.9° C	62%	29.96 in/ 1014.4 hPa	9.0 miles/ 14.5 km	9.2 mph/ 14.8 km/h South	Overcast
11:42 AM	60.8° F/16.0° C	46.4° F/8.0° C	59%	29.93 in/ 1013.4 hPa	10.0 miles/ 16.1 km	9.2 mph/ 14.8 km/h South	Overcast

VEHICLES

Vehicle One (VI, 1992 Saturn SL-1)

Year:	1992
Make:	Saturn
Model:	SL-1
Color:	Brown
License:	[redacted]
VIN:	[redacted]
Engine:	1.9L 85 hp 14
Transmission:	4 speed automatic
Weight:	2313 pounds

Damage Description:

Front:
V1 sustained no visible damage to this end.

Left:
The majority of the damage sustained by V1 occurred on this side. At the deepest intrusion, the crush measured 36". The entire left side from the driver's side door to the passenger side door was crushed. There was minor to moderate crushing to the front and rear quarter panels.

Right:
Besides the removing of the passenger front and rear doors by SLFD with the Jaws of Life, no visible collision damage was observed on this side.

VEHICLES (Continued)

MAIT PREPARER'S NAME	I.D. NUMBER	DATE	REVIEWER'S NAME	DATE
A. Ahubudu	1120	1/11/07		

DATE OF INCIDENT	TIME	NCIC NUMBER	OFFICER I.D.	NUMBER
Jan 11, 2006	1057	3710	1120	07-000883

Damage (Continued)

Rear:
V1 sustained no visible damage to this end.

Vehicle Two (V2,1995 Volkswagen Jetta III Celebration Edition)

Year:	1995
Make:	Volkswagen
Model:	Jetta III Celebration Edition
Color:	Black
License:	[redacted]
VIN:	[redacted]
Engine:	2.0 L 115 hp I4
Transmission:	5 Speed manual
Weight:	2648 pounds

Damage Description:

Front:
The majority of the damage sustained by V2 was isolated to this side. There was moderate crushing to the front bumper and hood. The length of crush to the front end was 6" at its deepest point. The both headlight and lighting assemblies were broken and knocked out.

Left:
Besides minor crush and dents to the front portion of the front quarter panel, no other visible damage was observed to this side.

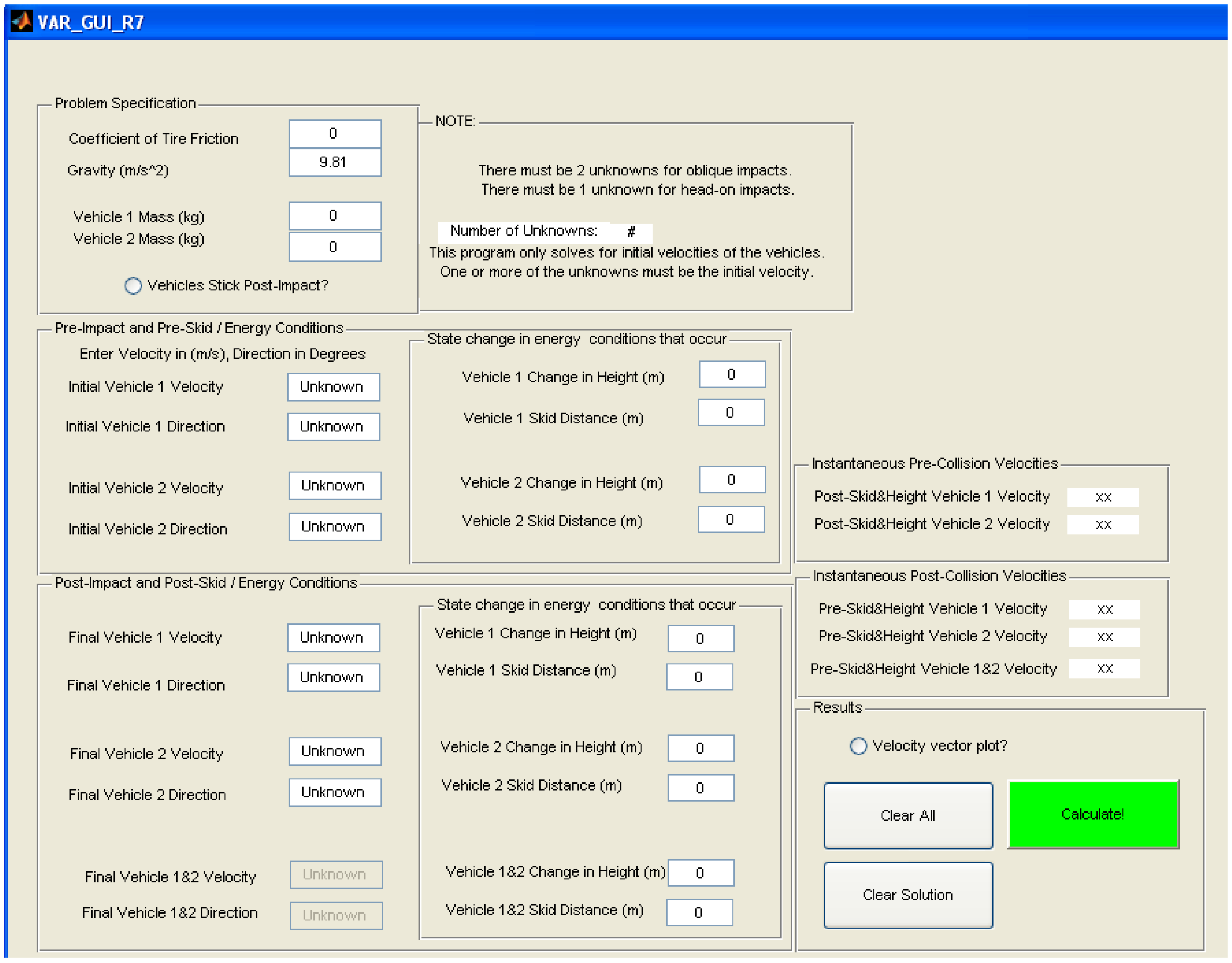
Right:
Besides minor crush and dents to the front portion of the front quarter panel, no other visible damage was observed to this side.

Rear:
V2 sustained no visible damage to this end.

MAIT PREPARER'S NAME	I.D. NUMBER	DATE	REVIEWER'S NAME	DATE
A. Ahubudu	1120	1/11/07		

Appendix F MatLab GUI User Guide

This guide serves as an overview on how to use the VAR GUI to solve for two vehicle impacts. Figure 45 below shows the input window of the MatLab GUI upon startup.



VAR_GUI_R7

Problem Specification

Coefficient of Tire Friction: 0

Gravity (m/s^2): 9.81

Vehicle 1 Mass (kg): 0

Vehicle 2 Mass (kg): 0

Vehicles Stick Post-Impact?

NOTE:

There must be 2 unknowns for oblique impacts.
There must be 1 unknown for head-on impacts.

Number of Unknowns: #

This program only solves for initial velocities of the vehicles.
One or more of the unknowns must be the initial velocity.

Pre-Impact and Pre-Skid / Energy Conditions

Enter Velocity in (m/s), Direction in Degrees

Initial Vehicle 1 Velocity: Unknown

Initial Vehicle 1 Direction: Unknown

Initial Vehicle 2 Velocity: Unknown

Initial Vehicle 2 Direction: Unknown

State change in energy conditions that occur

Vehicle 1 Change in Height (m): 0

Vehicle 1 Skid Distance (m): 0

Vehicle 2 Change in Height (m): 0

Vehicle 2 Skid Distance (m): 0

Instantaneous Pre-Collision Velocities

Post-Skid&Height Vehicle 1 Velocity: xx

Post-Skid&Height Vehicle 2 Velocity: xx

Post-Impact and Post-Skid / Energy Conditions

Final Vehicle 1 Velocity: Unknown

Final Vehicle 1 Direction: Unknown

Final Vehicle 2 Velocity: Unknown

Final Vehicle 2 Direction: Unknown

Final Vehicle 1&2 Velocity: Unknown

Final Vehicle 1&2 Direction: Unknown

State change in energy conditions that occur

Vehicle 1 Change in Height (m): 0

Vehicle 1 Skid Distance (m): 0

Vehicle 2 Change in Height (m): 0

Vehicle 2 Skid Distance (m): 0

Vehicle 1&2 Change in Height (m): 0

Vehicle 1&2 Skid Distance (m): 0

Instantaneous Post-Collision Velocities

Pre-Skid&Height Vehicle 1 Velocity: xx

Pre-Skid&Height Vehicle 2 Velocity: xx

Pre-Skid&Height Vehicle 1&2 Velocity: xx

Results

Velocity vector plot?

Figure 45. MatLab GUI at startup.

Currently Supported Scenarios

The MatLab code will automatically calculate for parameters left as "unknown", for the supported scenarios shown in Table 8 and Table 9. These tables illustrate the known and unknown parameters of the problem indicated by "1" and "0", respectively. V1, V2, and V12 indicate velocities of vehicle 1, vehicle 2, and vehicle 1&2 stuck, respectively.

Table 8. Supported cases for 2 vehicle collisions where vehicles do not stick together post-impact.

	Pre-Collision				Post-Collision			
	V1		V2		V1		V2	
Case	Mag	Dir	Mag	Dir	Mag	Dir	Mag	Dir
A	1	1	0	0	1	1	1	1
G	1	1	0	1	1	1	1	0
D	1	1	0	1	1	1	0	1
F	1	0	0	1	1	1	1	1
C	0	1	0	1	1	1	1	1
E	1	1	0	1	1	1	1	1

Table 9. Supported cases for 2 vehicle collisions where vehicles stick together post-impact.

	Pre-Collision				Post-Collision	
	V1		V2		V12	
Case	Mag	Dir	Mag	Dir	Mag	Dir
A	1	1	0	0	1	1
B	1	1	0	1	0	1
G	1	1	0	1	1	0
F	1	0	0	1	1	1
C	0	1	0	1	1	1
E	1	1	0	1	1	1

Syntax

Angle Convention

All angles should be inputted with North as the zero angle reference point, east as 90°, south as 180°, and west as 270°. For example, a vehicle traveling southwest would correspond to a direction of 225°, shown below.

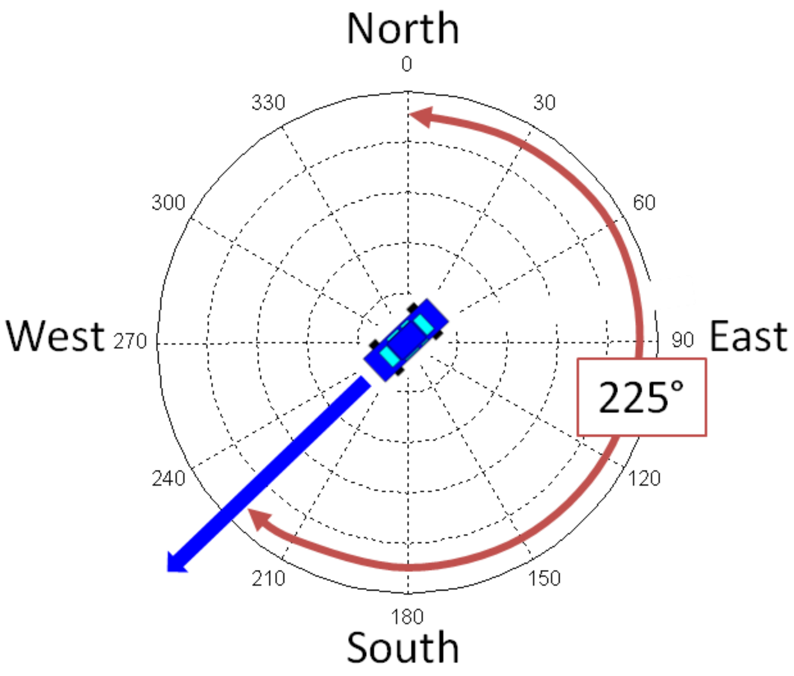


Figure 46. Angle dimension for vehicle traveling in southwest direction. The arrow shown in blue represents the direction vector of the vehicle. The red dimension indicates the corresponding angle.

Entering Known Parameters

The program is set up to accept all user inputs as summarized in the figure below. This section will provide a quick overview on the sign convention and entry of the parameters.

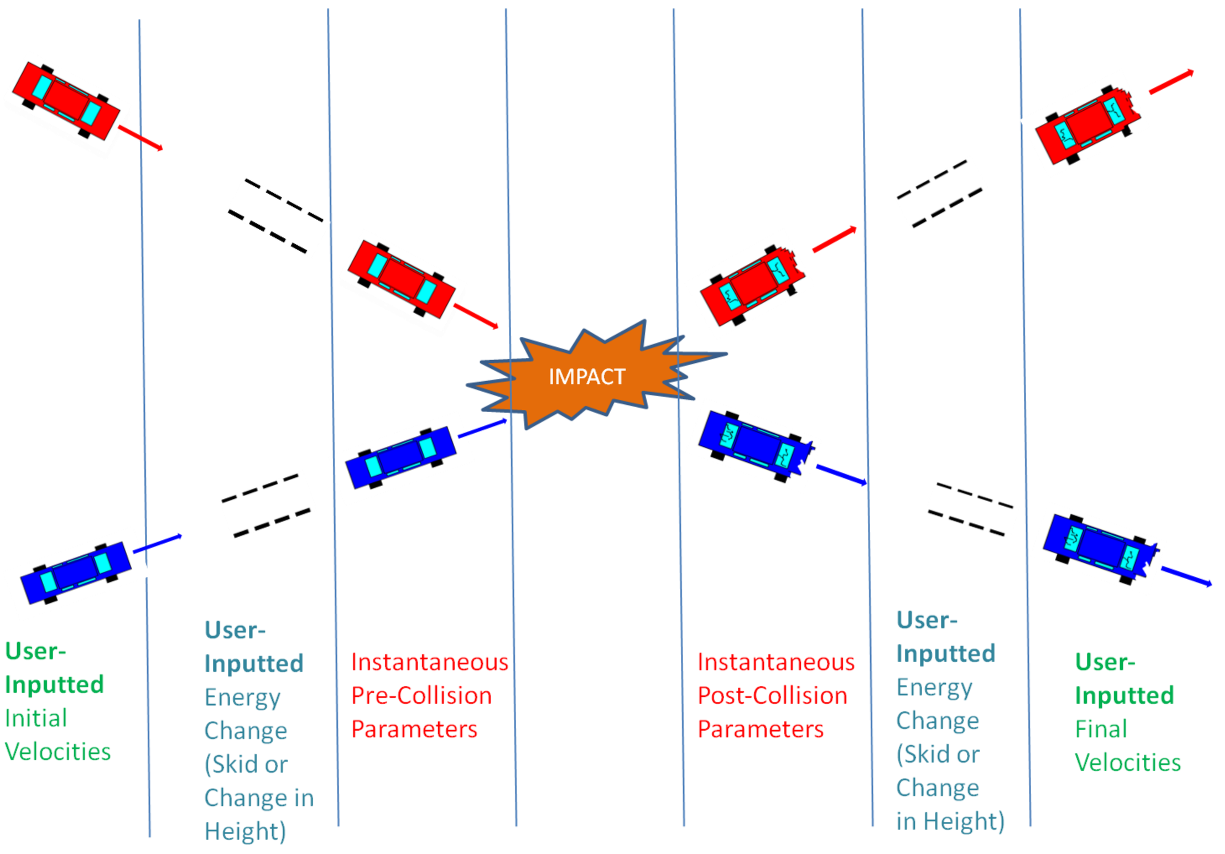

IMPACT
User-Inputted Initial Velocities
User-Inputted Energy Change (Skid or Change in Height)
Instantaneous Pre-Collision Parameters
Instantaneous Post-Collision Parameters
User-Inputted Energy Change (Skid or Change in Height)
User-Inputted Final Velocities

Figure 47. User-input syntax for VAR GUI.

Pre-Collision Velocity and Changes in Height

The MatLab program refers to all entered initial velocities as pre-skid and pre-change-in-height velocities. That is, it assumes all pre-collision velocities are stated before any skidding or change in height has occurred. When the user inputs a pre-skid initial velocity, skid distance, and change in height, MatLab will calculate the post-skid and post-change-in-height initial velocity as an intermediate step. If the post-skid/post-change-in-height initial velocity is known, simply input zero for skid distance and zero for height. A positive change in height indicates a vehicle has increased in elevation before impact, a negative implies it has descended.

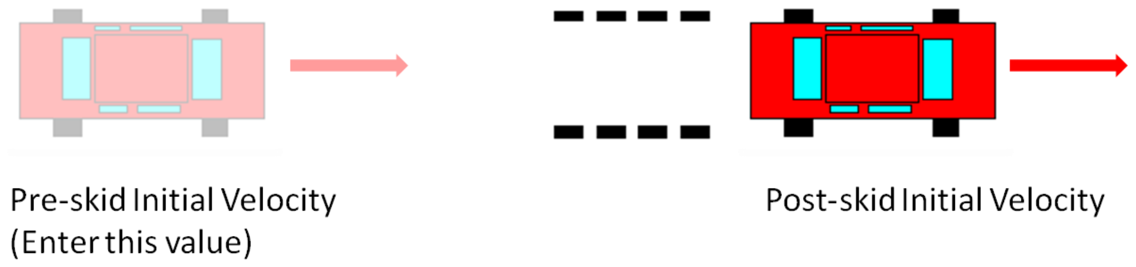


Figure 48. Convention for entering pre-collision velocity.

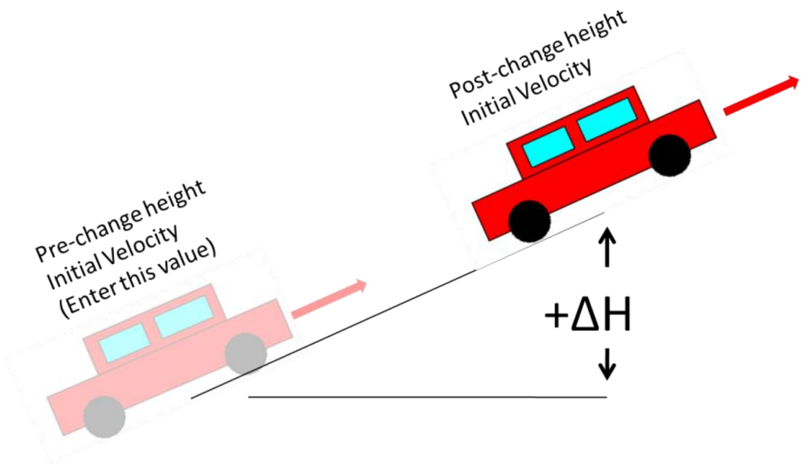


Figure 49. Convention for entering a pre-collision increase in height.

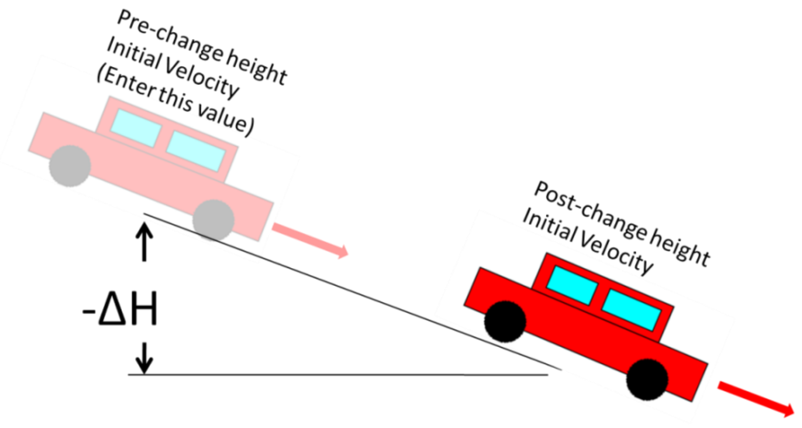


Figure 50. Convention for entering a pre-collision decrease in height.

Post-Collision Velocity

The program refers to all entered final velocities as post-skid velocities. That is, it assumes all user-entered post-collision velocities are stated <u>after</u> any skidding or change in height has occurred. If the pre-skid/pre-change-in-height final velocity is known, simply input zero for skid distance and height.

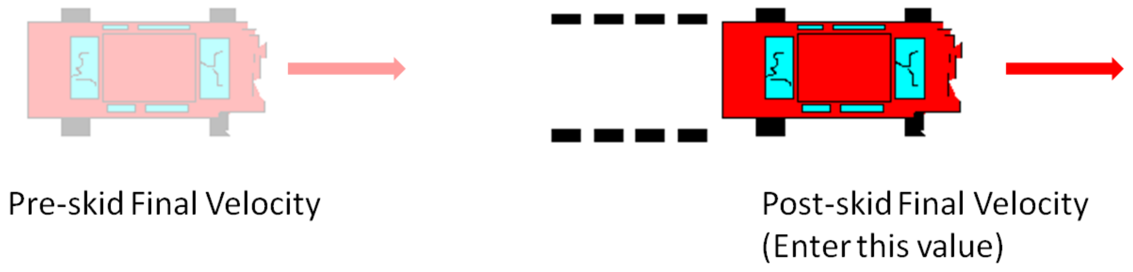


Figure 51. Convention for entering post-collision velocity.

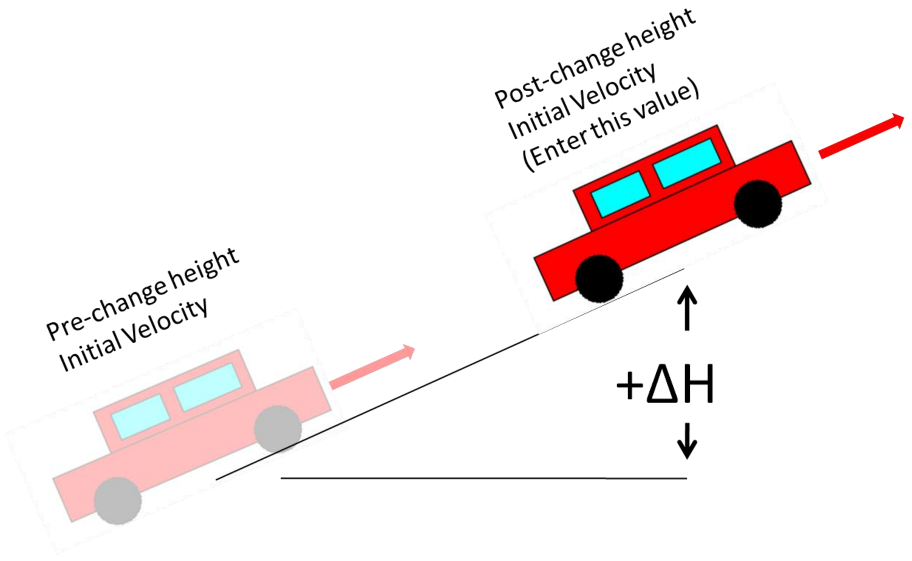


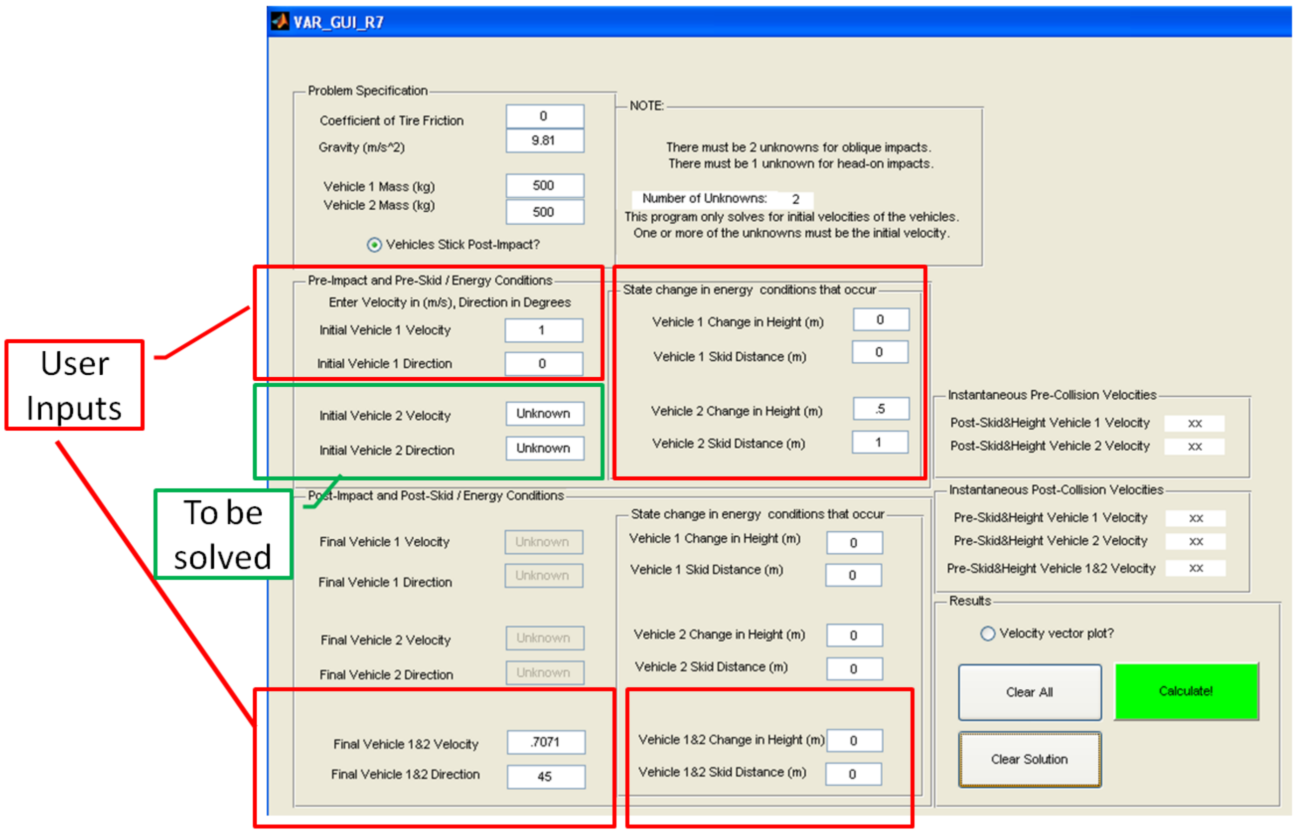
Figure 52. Convention for entering a post-collision change in height. (same as pre-collision)

Reasoning

In order to apply momentum laws for our collision, we need to know the instantaneous velocities of the vehicles right before and right after the impact. These velocities correspond to the post-skid and post-change-in-height initial velocities and the pre-skid and pre-change-in-height final velocities. By setting up the inputs as shown, the code can automatically take into account skid distances when solving for unknown velocities. Essentially, this allows us to account for any energy loss/gain between the inputted velocities and the instantaneous velocities before or after impact.

Example

Shown in Figure 53 is the MatLab GUI window with all user-inputs entered. The variables to be solved are left as “unknowns” and will be automatically solved for.



VAR_GUI_R7

Problem Specification

Coefficient of Tire Friction	0
Gravity (m/s^2)	9.81
Vehicle 1 Mass (kg)	500
Vehicle 2 Mass (kg)	500

[✓ Vehicles Stick Post-Impact?]

NOTE:

There must be 2 unknowns for oblique impacts.
There must be 1 unknown for head-on impacts.

Number of Unknowns: 2

This program only solves for initial velocities of the vehicles.
One or more of the unknowns must be the initial velocity.

Pre-Impact and Pre-Skid / Energy Conditions

Enter Velocity in (m/s), Direction in Degrees

Initial Vehicle 1 Velocity	1
Initial Vehicle 1 Direction	0
Initial Vehicle 2 Velocity	Unknown
Initial Vehicle 2 Direction	Unknown

State change in energy conditions that occur

Vehicle 1 Change in Height (m)	0
Vehicle 1 Skid Distance (m)	0
Vehicle 2 Change in Height (m)	.5
Vehicle 2 Skid Distance (m)	1

Instantaneous Pre-Collision Velocities

Post-Skid&Height Vehicle 1 Velocity	xx
Post-Skid&Height Vehicle 2 Velocity	xx

Post-Impact and Post-Skid / Energy Conditions

Final Vehicle 1 Velocity	Unknown
Final Vehicle 1 Direction	Unknown
Final Vehicle 2 Velocity	Unknown
Final Vehicle 2 Direction	Unknown
Final Vehicle 1&2 Velocity	.7071
Final Vehicle 1&2 Direction	45

State change in energy conditions that occur

Vehicle 1 Change in Height (m)	0
Vehicle 1 Skid Distance (m)	0
Vehicle 2 Change in Height (m)	0
Vehicle 2 Skid Distance (m)	0
Vehicle 1&2 Change in Height (m)	0
Vehicle 1&2 Skid Distance (m)	0

Instantaneous Post-Collision Velocities

Pre-Skid&Height Vehicle 1 Velocity	xx
Pre-Skid&Height Vehicle 2 Velocity	xx
Pre-Skid&Height Vehicle 1&2 Velocity	xx

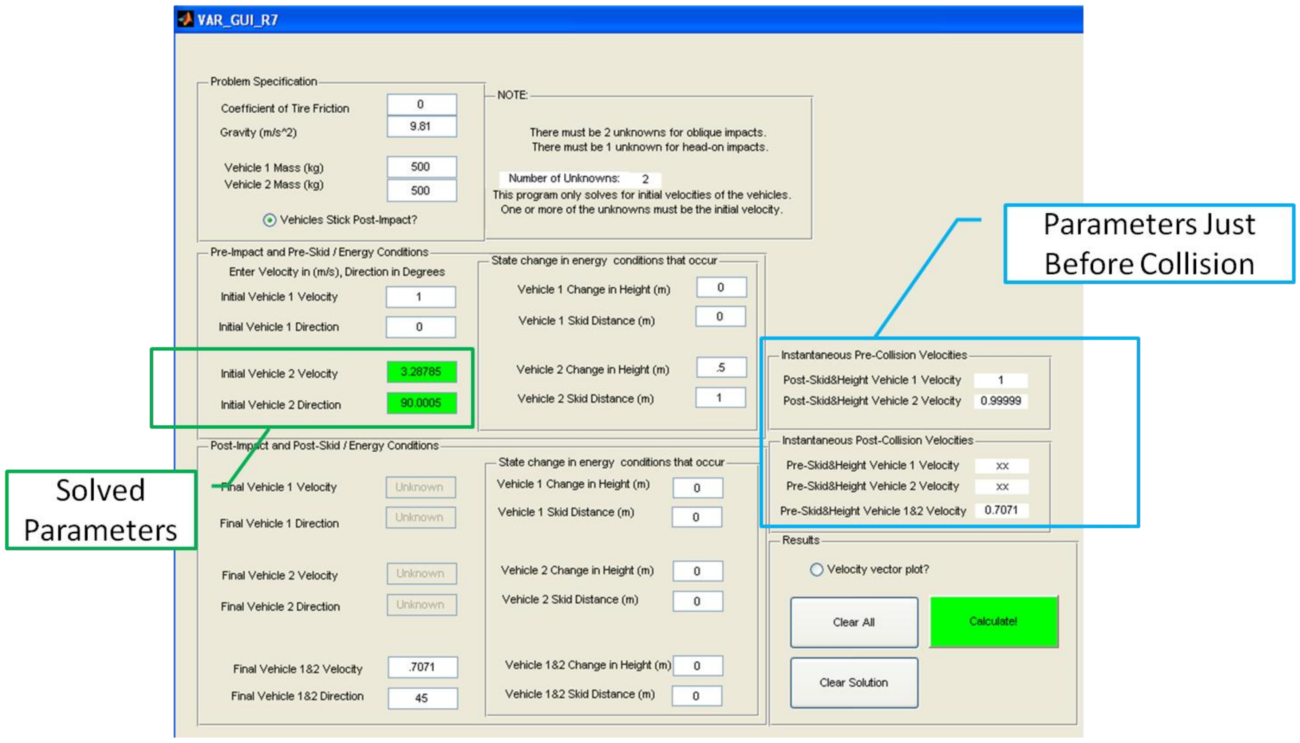
Results

Velocity vector plot?

Clear All | Calculate! | Clear Solution

Figure 53. Input window prior to running.

Shown in Figure 54 is the MatLab GUI window right after “Calculate!” is pressed. Highlighted in green are the solved variables. Highlighted in blue are the variables just before and after the collision occurs, as explained in Figure 47.



Problem Specification

Coefficient of Tire Friction	0
Gravity (m/s^2)	9.81
Vehicle 1 Mass (kg)	500
Vehicle 2 Mass (kg)	500

[✓ Vehicles Stick Post-Impact?]

NOTE:

There must be 2 unknowns for oblique impacts.
There must be 1 unknown for head-on impacts.

Number of Unknowns: 2

This program only solves for initial velocities of the vehicles.
One or more of the unknowns must be the initial velocity.

Pre-Impact and Pre-Skid / Energy Conditions

Enter Velocity in (m/s), Direction in Degrees

Initial Vehicle 1 Velocity	1
Initial Vehicle 1 Direction	0
Initial Vehicle 2 Velocity	3.28785
Initial Vehicle 2 Direction	90.0005

State change in energy conditions that occur

Vehicle 1 Change in Height (m)	0
Vehicle 1 Skid Distance (m)	0
Vehicle 2 Change in Height (m)	.5
Vehicle 2 Skid Distance (m)	1

Instantaneous Pre-Collision Velocities

Post-Skid&Height Vehicle 1 Velocity	1
Post-Skid&Height Vehicle 2 Velocity	0.99999

Instantaneous Post-Collision Velocities

Pre-Skid&Height Vehicle 1 Velocity	xx
Pre-Skid&Height Vehicle 2 Velocity	xx
Pre-Skid&Height Vehicle 1&2 Velocity	0.7071

Post-Impact and Post-Skid / Energy Conditions

Final Vehicle 1 Velocity	Unknown
Final Vehicle 1 Direction	Unknown
Final Vehicle 2 Velocity	Unknown
Final Vehicle 2 Direction	Unknown
Final Vehicle 1&2 Velocity	.7071
Final Vehicle 1&2 Direction	45

State change in energy conditions that occur

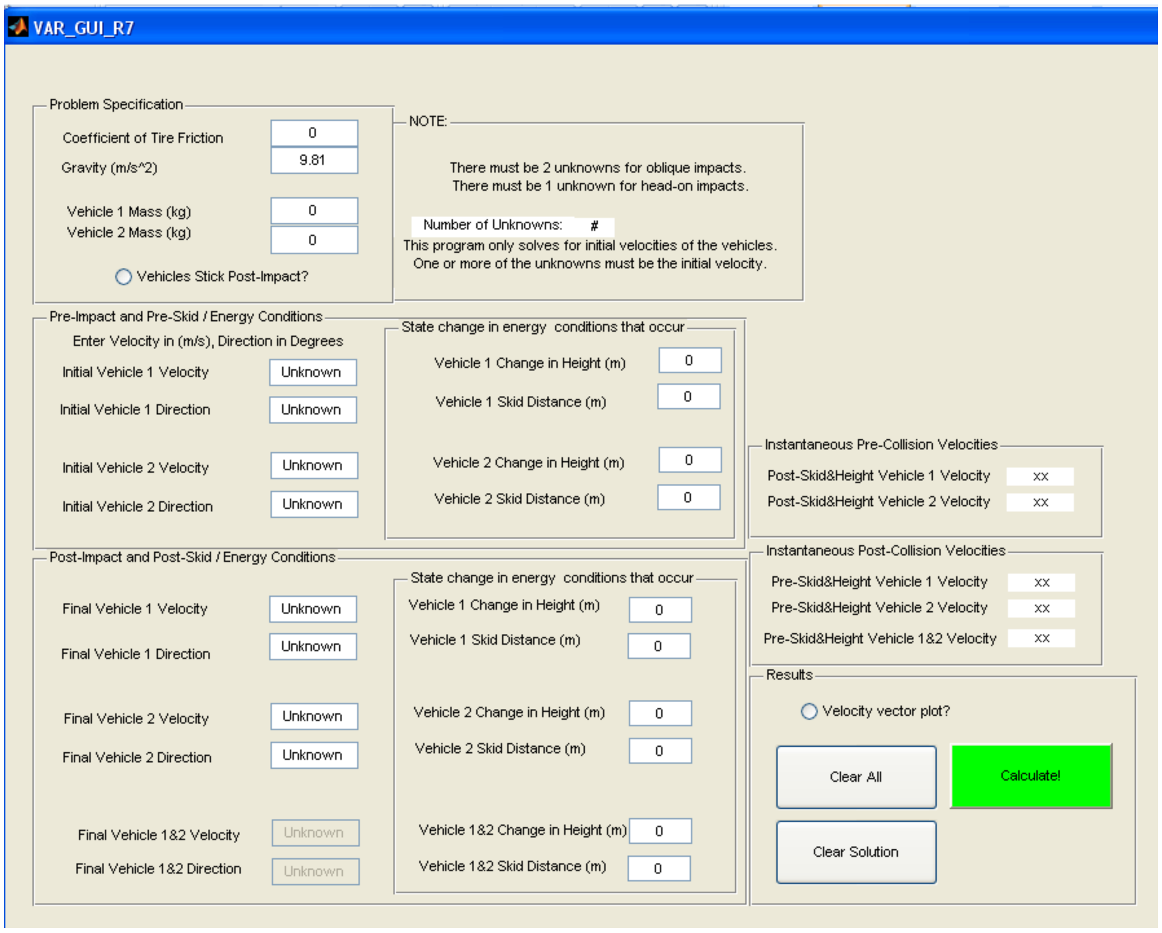
Vehicle 1 Change in Height (m)	0
Vehicle 1 Skid Distance (m)	0
Vehicle 2 Change in Height (m)	0
Vehicle 2 Skid Distance (m)	0
Vehicle 1&2 Change in Height (m)	0
Vehicle 1&2 Skid Distance (m)	0

Results

Velocity vector plot?

Figure 54. Input window with calculated results.

Appendix G   VAR MatLab GUI



VAR_GUI_R7

Problem Specification

Coefficient of Tire Friction	0
Gravity (m/s^2)	9.81
Vehicle 1 Mass (kg)	0
Vehicle 2 Mass (kg)	0

Vehicles Stick Post-Impact?

NOTE:

There must be 2 unknowns for oblique impacts.
There must be 1 unknown for head-on impacts.

Number of Unknowns: #

This program only solves for initial velocities of the vehicles.
One or more of the unknowns must be the initial velocity.

Pre-Impact and Pre-Skid / Energy Conditions

Enter Velocity in (m/s), Direction in Degrees

Initial Vehicle 1 Velocity	Unknown
Initial Vehicle 1 Direction	Unknown
Initial Vehicle 2 Velocity	Unknown
Initial Vehicle 2 Direction	Unknown

State change in energy conditions that occur

Vehicle 1 Change in Height (m)	0
Vehicle 1 Skid Distance (m)	0
Vehicle 2 Change in Height (m)	0
Vehicle 2 Skid Distance (m)	0

Instantaneous Pre-Collision Velocities

Post-Skid&Height Vehicle 1 Velocity	xx
Post-Skid&Height Vehicle 2 Velocity	xx

Post-Impact and Post-Skid / Energy Conditions

Final Vehicle 1 Velocity	Unknown
Final Vehicle 1 Direction	Unknown
Final Vehicle 2 Velocity	Unknown
Final Vehicle 2 Direction	Unknown
Final Vehicle 1&2 Velocity	Unknown
Final Vehicle 1&2 Direction	Unknown

State change in energy conditions that occur

Vehicle 1 Change in Height (m)	0
Vehicle 1 Skid Distance (m)	0
Vehicle 2 Change in Height (m)	0
Vehicle 2 Skid Distance (m)	0
Vehicle 1&2 Change in Height (m)	0
Vehicle 1&2 Skid Distance (m)	0

Instantaneous Post-Collision Velocities

Pre-Skid&Height Vehicle 1 Velocity	xx
Pre-Skid&Height Vehicle 2 Velocity	xx
Pre-Skid&Height Vehicle 1&2 Velocity	xx

Results

Velocity vector plot?

```
%VAR MEA 2-Vehicle Collision Solver
%Cal Poly State University, San Luis Obispo
%Lawrence Fong (lhfong@calpoly.edu)


function varargout = VAR_GUI_R7(varargin)
% VAR_GUI_R7 M-file for VAR_GUI_R7.fig
%      VAR_GUI_R7, by itself, creates a new VAR_GUI_R7 or raises the existing
%      singleton*.
%
%      H = VAR_GUI_R7 returns the handle to a new VAR_GUI_R7 or the handle to
%      the existing singleton*.
%
%      VAR_GUI_R7('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in VAR_GUI_R7.M with the given input
%      arguments.
%
%      VAR_GUI_R7('Property','Value',...) creates a new VAR_GUI_R7 or
%      raises the
%      existing singleton*.  Starting from the left, property value pairs
%      are
%      applied to the GUI before VAR_GUI_R7_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
%      application
%      stop.  All inputs are passed to VAR_GUI_R7_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help VAR_GUI_R7

% Last Modified by GUIDE v2.5 09-Aug-2009 15:10:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @VAR_GUI_R7_OpeningFcn, ...
                   'gui_OutputFcn',  @VAR_GUI_R7_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before VAR_GUI_R7 is made visible.
function VAR_GUI_R7_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to VAR_GUI_R7 (see VARARGIN)

% Choose default command line output for VAR_GUI_R7
handles.output = hObject;

% Update handles structure
    set(handles.V_f12, 'Enable', 'off');
    set(handles.V_f12D, 'Enable', 'off');
guidata(hObject, handles);

% UIWAIT makes VAR_GUI_R7 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = VAR_GUI_R7_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function V_f1_Callback(hObject, eventdata, handles)
% hObject    handle to V_f1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_f1 as text
%        str2double(get(hObject,'String')) returns contents of V_f1 as a
%        double

%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','Unknown')
end
%------------



% --- Executes during object creation, after setting all properties.
function V_f1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_f1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function V_f1D_Callback(hObject, eventdata, handles)
% hObject    handle to V_f1D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_f1D as text
%        str2double(get(hObject,'String')) returns contents of V_f1D as a
double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','Unknown')
end
%------------



% --- Executes during object creation, after setting all properties.
function V_f1D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_f1D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function V_f2_Callback(hObject, eventdata, handles)
% hObject    handle to V_f2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_f2 as text
%        str2double(get(hObject,'String')) returns contents of V_f2 as a
double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','Unknown')
```

```
end
%------------


% --- Executes during object creation, after setting all properties.
function V_f2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_f2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function V_f2D_Callback(hObject, eventdata, handles)
% hObject    handle to V_f2D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_f2D as text
%        str2double(get(hObject,'String')) returns contents of V_f2D as a
double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','Unknown')
end
%------------


% --- Executes during object creation, after setting all properties.
function V_f2D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_f2D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function V_o1_Callback(hObject, eventdata, handles)
% hObject    handle to V_o1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of V_o1 as text
%        str2double(get(hObject,'String')) returns contents of V_o1 as a
double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','Unknown')
end

%------------



% --- Executes during object creation, after setting all properties.
function V_o1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_o1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function V_o1D_Callback(hObject, eventdata, handles)
% hObject    handle to V_o1D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_o1D as text
%        str2double(get(hObject,'String')) returns contents of V_o1D as a
double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','Unknown')
end
%------------



% --- Executes during object creation, after setting all properties.
function V_o1D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_o1D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function V_o2_Callback(hObject, eventdata, handles)
% hObject    handle to V_o2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_o2 as text
%        str2double(get(hObject,'String')) returns contents of V_o2 as a
double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','Unknown')
end
%------------



% --- Executes during object creation, after setting all properties.
function V_o2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_o2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function V_o2D_Callback(hObject, eventdata, handles)
% hObject    handle to V_o2D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_o2D as text
%        str2double(get(hObject,'String')) returns contents of V_o2D as a
double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','Unknown')
end
%------------



% --- Executes during object creation, after setting all properties.
function V_o2D_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to V_o2D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function friction_Callback(hObject, eventdata, handles)
% hObject    handle to friction (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of friction as text
%        str2double(get(hObject,'String')) returns contents of friction as a
double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end
%------------


% --- Executes during object creation, after setting all properties.
function friction_CreateFcn(hObject, eventdata, handles)
% hObject    handle to friction (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function gravity_Callback(hObject, eventdata, handles)
% hObject    handle to gravity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of gravity as text
%        str2double(get(hObject,'String')) returns contents of gravity as a
double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
```

```
if (isempty(input))
     set(hObject,'String','9.81')
end
%-------------



% --- Executes during object creation, after setting all properties.
function gravity_CreateFcn(hObject, eventdata, handles)
% hObject    handle to gravity (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in stickYes.
function stickYes_Callback(hObject, eventdata, handles)
% hObject    handle to stickYes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of stickYes
if get(hObject,'Value')==1
    set(handles.V_f1,'String','Unknown');
    set(handles.V_f1D,'String','Unknown');
    set(handles.V_f2,'String','Unknown');
    set(handles.V_f2D,'String','Unknown');
    set(handles.V_f12,'String','Unknown');
    set(handles.V_f12D,'String','Unknown');


    set(handles.V_f12, 'Enable', 'on');
    set(handles.V_f12D, 'Enable', 'on');
    set(handles.V_f1, 'Enable', 'off');
    set(handles.V_f1D, 'Enable', 'off');
    set(handles.V_f2, 'Enable', 'off');
    set(handles.V_f2D, 'Enable', 'off');



else
    set(handles.V_f1,'String','Unknown');
    set(handles.V_f1D,'String','Unknown');
    set(handles.V_f2,'String','Unknown');
    set(handles.V_f2D,'String','Unknown');

    set(handles.V_f12,'String','Unknown');
    set(handles.V_f12D,'String','Unknown');

    set(handles.V_f12, 'Enable', 'off');
    set(handles.V_f12D, 'Enable', 'off');
    set(handles.V_f1, 'Enable', 'on');
    set(handles.V_f1D, 'Enable', 'on');
    set(handles.V_f2, 'Enable', 'on');
    set(handles.V_f2D, 'Enable', 'on');
end


function mass_1_Callback(hObject, eventdata, handles)
% hObject    handle to mass_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of mass_1 as text
%        str2double(get(hObject,'String')) returns contents of mass_1 as a
double
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end


% --- Executes during object creation, after setting all properties.
function mass_1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mass_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function mass_2_Callback(hObject, eventdata, handles)
% hObject    handle to mass_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of mass_2 as text
%        str2double(get(hObject,'String')) returns contents of mass_2 as a
double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end
%------------


% --- Executes during object creation, after setting all properties.
function mass_2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mass_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function V_f12_Callback(hObject, eventdata, handles)
% hObject    handle to V_f12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_f12 as text
%        str2double(get(hObject,'String')) returns contents of V_f12 as a
%        double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','Unknown')
end
%------------



% --- Executes during object creation, after setting all properties.
function V_f12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_f12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function V_f12D_Callback(hObject, eventdata, handles)
% hObject    handle to V_f12D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_f12D as text
%        str2double(get(hObject,'String')) returns contents of V_f12D as a
double
%Following code checks to make sure the input is a number
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','Unknown')
```

```
end
%------------


% --- Executes during object creation, after setting all properties.
function V_f12D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_f12D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function height1_f_Callback(hObject, eventdata, handles)
% hObject    handle to height1_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of height1_f as text
%        str2double(get(hObject,'String')) returns contents of height1_f as a
double
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end


% --- Executes during object creation, after setting all properties.
function height1_f_CreateFcn(hObject, eventdata, handles)
% hObject    handle to height1_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function skid1_f_Callback(hObject, eventdata, handles)
% hObject    handle to skid1_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of skid1_f as text
%        str2double(get(hObject,'String')) returns contents of skid1_f as a
double
```

```
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end


% --- Executes during object creation, after setting all properties.
function skid1_f_CreateFcn(hObject, eventdata, handles)
% hObject    handle to skid1_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function height2_f_Callback(hObject, eventdata, handles)
% hObject    handle to height2_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of height2_f as text
%        str2double(get(hObject,'String')) returns contents of height2_f as a
double
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end


% --- Executes during object creation, after setting all properties.
function height2_f_CreateFcn(hObject, eventdata, handles)
% hObject    handle to height2_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function skid2_f_Callback(hObject, eventdata, handles)
% hObject    handle to skid2_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of skid2_f as text
%        str2double(get(hObject,'String')) returns contents of skid2_f as a
double
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end


% --- Executes during object creation, after setting all properties.
function skid2_f_CreateFcn(hObject, eventdata, handles)
% hObject    handle to skid2_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function height12_f_Callback(hObject, eventdata, handles)
% hObject    handle to height12_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of height12_f as text
%        str2double(get(hObject,'String')) returns contents of height12_f as
a double
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end


% --- Executes during object creation, after setting all properties.
function height12_f_CreateFcn(hObject, eventdata, handles)
% hObject    handle to height12_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function skid12_f_Callback(hObject, eventdata, handles)
% hObject    handle to skid12_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of skid12_f as text
%        str2double(get(hObject,'String')) returns contents of skid12_f as a
double
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end


% --- Executes during object creation, after setting all properties.
function skid12_f_CreateFcn(hObject, eventdata, handles)
% hObject    handle to skid12_f (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function height1_o_Callback(hObject, eventdata, handles)
% hObject    handle to height1_o (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of height1_o as text
%        str2double(get(hObject,'String')) returns contents of height1_o as a
double
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end


% --- Executes during object creation, after setting all properties.
function height1_o_CreateFcn(hObject, eventdata, handles)
% hObject    handle to height1_o (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function skid1_o_Callback(hObject, eventdata, handles)
% hObject    handle to skid1_o (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of skid1_o as text
%        str2double(get(hObject,'String')) returns contents of skid1_o as a
double
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end


% --- Executes during object creation, after setting all properties.
function skid1_o_CreateFcn(hObject, eventdata, handles)
% hObject    handle to skid1_o (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function height2_o_Callback(hObject, eventdata, handles)
% hObject    handle to height2_o (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of height2_o as text
%        str2double(get(hObject,'String')) returns contents of height2_o as a
double
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end


% --- Executes during object creation, after setting all properties.
function height2_o_CreateFcn(hObject, eventdata, handles)
% hObject    handle to height2_o (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function skid2_o_Callback(hObject, eventdata, handles)
% hObject    handle to skid2_o (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of skid2_o as text
%        str2double(get(hObject,'String')) returns contents of skid2_o as a
double
input = str2num(get(hObject,'String'));
if (isempty(input))
     set(hObject,'String','0')
end



% --- Executes during object creation, after setting all properties.
function skid2_o_CreateFcn(hObject, eventdata, handles)
% hObject    handle to skid2_o (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in velocityPlot.
function velocityPlot_Callback(hObject, eventdata, handles)
% hObject    handle to velocityPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% Hint: get(hObject,'Value') returns toggle state of velocityPlot
% --- Executes on button press in clearSolution.
function clearSolution_Callback(hObject, eventdata, handles)
% hObject    handle to clearSolution (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if isequal(get(handles.V_o2,'BackgroundColor'),[1 1 1])
else
    set(handles.V_o2,'BackgroundColor','white')
    set(handles.V_o2,'String','Unknown')
end
```

```
if isequal(get(handles.V_o1,'BackgroundColor'),[1 1 1])
else
    set(handles.V_o1,'BackgroundColor','white')
    set(handles.V_o1,'String','Unknown')
end

if isequal(get(handles.V_o1D,'BackgroundColor'),[1 1 1])
else
    set(handles.V_o1D,'BackgroundColor','white')
    set(handles.V_o1D,'String','Unknown')
end

if isequal(get(handles.V_o2,'BackgroundColor'),[1 1 1])
else
    set(handles.V_o2,'String','Unknown')
    set(handles.V_o2,'BackgroundColor','white')
end

if isequal(get(handles.V_o2D,'BackgroundColor'),[1 1 1])
else
set(handles.V_o2D,'String','Unknown')
set(handles.V_o2D,'BackgroundColor','white')
end

if isequal(get(handles.V_f1,'BackgroundColor'),[1 1 1])
else
set(handles.V_f1,'String','Unknown')
set(handles.V_f1,'BackgroundColor','white')
end

if isequal(get(handles.V_f1D,'BackgroundColor'),[1 1 1])
else
set(handles.V_f1D,'String','Unknown')
set(handles.V_f1D,'BackgroundColor','white')
end

if isequal(get(handles.V_f2,'BackgroundColor'),[1 1 1])
else
set(handles.V_f2,'String','Unknown')
set(handles.V_f2,'BackgroundColor','white')
end

if isequal(get(handles.V_f2D,'BackgroundColor'),[1 1 1])
else
set(handles.V_f2D,'String','Unknown')
set(handles.V_f2D,'BackgroundColor','white')
end

if isequal(get(handles.V_f12,'BackgroundColor'),[1 1 1])
else
set(handles.V_f12,'String','Unknown')
set(handles.V_f12,'BackgroundColor','white')
end
```

```
if isequal(get(handles.V_f12D,'BackgroundColor'),[1 1 1])
else
set(handles.V_f12D,'String','Unknown')
set(handles.V_f12D,'BackgroundColor','white')
end

set(handles.V_o1_c,'String','xx')
set(handles.V_o2_c,'String','xx')
set(handles.V_f1_c,'String','xx')
set(handles.V_f2_c,'String','xx')
set(handles.V_f12_c,'String','xx')






% --- Executes on button press in clearAll.
function clearAll_Callback(hObject, eventdata, handles)
% hObject    handle to clearAll (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.V_o1,'BackgroundColor','white')
set(handles.V_o1,'String','Unknown')

set(handles.V_o1D,'BackgroundColor','white')
set(handles.V_o1D,'String','Unknown')

set(handles.V_o2,'String','Unknown')
set(handles.V_o2,'BackgroundColor','white')

set(handles.V_o2D,'String','Unknown')
set(handles.V_o2D,'BackgroundColor','white')

set(handles.V_f1,'String','Unknown')
set(handles.V_f1,'BackgroundColor','white')

set(handles.V_f1D,'String','Unknown')
set(handles.V_f1D,'BackgroundColor','white')

set(handles.V_f2,'String','Unknown')
set(handles.V_f2,'BackgroundColor','white')

set(handles.V_f2D,'String','Unknown')
set(handles.V_f2D,'BackgroundColor','white')


set(handles.V_f12,'String','Unknown')
set(handles.V_f12,'BackgroundColor','white')

set(handles.V_f12D,'String','Unknown')
set(handles.V_f12D,'BackgroundColor','white')

set(handles.height1_o,'String','0')
set(handles.height2_o,'String','0')
set(handles.height1_f,'String','0')
set(handles.height2_f,'String','0')
set(handles.height12_f,'String','0')

set(handles.skid1_f,'String','0')




% --- THIS PART IS THE PROGRAM ----------
function pushbutton_calc_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_calc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Reset all values in GUI
clc

stick = get(handles.stickYes,'Value');


set(handles.V_o1,'BackgroundColor','white')
set(handles.V_o1D,'BackgroundColor','white')
set(handles.V_o2,'BackgroundColor','white')
set(handles.V_o2D,'BackgroundColor','white')

if stick == 0
    set(handles.V_f1,'BackgroundColor','white')
    set(handles.V_f1D,'BackgroundColor','white')
    set(handles.V_f2,'BackgroundColor','white')
    set(handles.V_f2D,'BackgroundColor','white')
elseif stick ==1
    set(handles.V_f12,'BackgroundColor','white')
    set(handles.V_f12D,'BackgroundColor','white')
end

%Resets fields
set(handles.V_o1_c,'String','xx')
set(handles.V_o2_c,'String','xx')
set(handles.V_f1_c,'String','xx')
set(handles.V_f2_c,'String','xx')
set(handles.V_f12_c,'String','xx')




%% **********************
%Variable Initializations
%------------------------
%-------(DO NOT MODIFY)-------------
v1i_ask = 0;           %is magnitude of initial velocity1 known? (0 || 1)
theta1i_ask = 0;        %is direction of initial velocity1 known? (0 || 1)
v2i_ask = 0;           %is magnitude of initial velocity2 known? (0 || 1)
theta2i_ask = 0;        %is direction of initial velocity2 known? (0 || 1)
v12f_ask = 0;           %is magnitude of final velocity12 known? (0 || 1)
```

```
theta12f_ask = 0;        %is direction of final velocity12 known? (0 || 1)
v1f_ask = 0;            %is magnitude of final velocity1 known? (0 || 1)
theta1f_ask = 0;         %is direction of final velocity1 known? (0 || 1)
v2f_ask = 0;            %is magnitude of final velocity2 known? (0 || 1)
theta2f_ask = 0;         %is direction of final velocity2 known? (0 || 1)
height1=0;                         %What is the pre-collision change in
height of vehicle 1
v1i = 0;              %What is the pre-collision velocity magnitude for
vehicle 1
theta1i = 0;           %What is the pre-collision velocity direction for
vehicle 1
height2=0;                         %What is the pre-collision change in
height of vehicle 2
v2i = 0;              %What is the pre-collision velocity magnitude for
vehicle 2
theta2i = 0;           %What is the pre-collision velocity direction for
vehicle 2
height12=0;                        %What is the post-collision change in
height of vehicle 1&2 (Stuck together)
v12f = 0;               %What is the post-collision velocity magnitude for
vehicle 1&2 (Stuck together)
theta12f = 0;            %What is the post-collision velocity direction for
vehicle 1&2 (Stuck together)
heightFinal1=0;                    %What is the post-collision change in
height of vehicle 1
v1f = 0;                %What is the post-collision velocity magnitude for
vehicle 1
theta1f = 0;             %What is the post-collision velocity direction for
vehicle 1
heightFinal2=0;                    %What is the post-collision change in
height of vehicle 2
v2f = 0;                %What is the post-collision velocity magnitude for
vehicle 2
theta2f = 0;             %What is the post-collision velocity direction for
vehicle 2
changeHeight = 0;                   %is there a change in height of the
vehicles?
P1_i = 0;                           %Pre-collision Momentum for vehicle 1
P2_i = 0;                           %Pre-collision Momentum for vehicle 2
P12_f = 0;                          %Post-collision Momentum for vehicle 1&2
(stuck together)
P1_f = 0;                           %Post-collision Momentum for vehicle 1
(stuck together)
P2_f = 0;                           %Post-collision Momentum for vehicle 2
(stuck together)
validCase = 0;  %Checks to see if this is a supported scenario
%Instanteous pre/post collision velocities
v1i_c=0;
v2i_c=0;
v1f_c=0;
v2f_c=0;
v12f_c=0;

converged = 0; %Sets up variable for iteration cases
%END (DO NOT MODIFY)
```

```
oneVehicle = 0; %Temporary holder for test cases
headOn = 0;


g = str2num(get(handles.gravity,'String'));  %gravity (m/s^2)
u_k= str2num(get(handles.friction,'String')); %coefficient of friction for
tires
m1 = str2num(get(handles.mass_1,'String'));  %mass of vehicle 1
m2 = str2num(get(handles.mass_2,'String'));  %mass of vehicle 2
m12=m1+m2;                               %mass of vehicle 1 + mass of vehicle 2
(for stuck collisions)

if m1==0 || m2==0
    error('No masses inputted')
end


unknownCount = 0;
%--Retrieves Entered information about Vehicle 1 Velocity--
input = str2num(get(handles.V_o1,'String'));
if (isempty(input))
    v1i_ask = 0;
    unknownCount = unknownCount + 1;
else
    v1i_ask = 1;
    v1i = str2num(get(handles.V_o1,'String'));
end

input = str2num(get(handles.V_o1D,'String'));
if (isempty(input))
    theta1i_ask = 0;
    unknownCount = unknownCount + 1;
else
    theta1i_ask = 1;
    theta1i = str2num(get(handles.V_o1D,'String'));
end

if stick == 0
    input = str2num(get(handles.V_f1,'String'));
    if (isempty(input))
        v1f_ask = 0;
        unknownCount = unknownCount + 1;
    else
        v1f_ask = 1;
        v1f = str2num(get(handles.V_f1,'String'));
    end

    input = str2num(get(handles.V_f1D,'String'));
    if (isempty(input))
        theta1f_ask = 0;
        unknownCount = unknownCount + 1;
    else
        theta1f_ask = 1;
        theta1f = str2num(get(handles.V_f1D,'String'));
    end
end

%Retrieves Entered information about Vehicle 2 Velocity
input = str2num(get(handles.V_o2,'String'));
if (isempty(input))
    v2i_ask = 0;
    unknownCount = unknownCount + 1;
else
    v2i_ask = 1;
    v2i = str2num(get(handles.V_o2,'String'));
end

input = str2num(get(handles.V_o2D,'String'));
if (isempty(input))
    theta2i_ask = 0;
    unknownCount = unknownCount + 1;
else
    theta2i_ask = 1;
    theta2i = str2num(get(handles.V_o2D,'String'));
end

if stick == 0
    input = str2num(get(handles.V_f2,'String'));
    if (isempty(input))
        v2f_ask = 0;
        unknownCount = unknownCount + 1;
    else
        v2f_ask = 1;
        v2f = str2num(get(handles.V_f2,'String'));
    end

    input = str2num(get(handles.V_f2D,'String'));
    if (isempty(input))
        theta2f_ask = 0;
        unknownCount = unknownCount + 1;
    else
        theta2f_ask = 1;
        theta2f = str2num(get(handles.V_f2D,'String'));
    end
else

end

if stick == 1
    input = str2num(get(handles.V_f12,'String'));
    if (isempty(input))
        v12f_ask = 0;
        unknownCount = unknownCount + 1;
    else
        v12f_ask = 1;
        v12f = str2num(get(handles.V_f12,'String'));
    end

    input = str2num(get(handles.V_f12D,'String'));
    if (isempty(input))
        theta12f_ask = 0;
        unknownCount = unknownCount + 1;
    else
        theta12f_ask = 1;
        theta12f = str2num(get(handles.V_f12D,'String'));
    end
end

set(handles.numberUnknowns,'String',unknownCount);
%*****************************************


if (theta1i_ask == 1 && theta2i_ask == 1) && (theta1i == (theta2i + 180) ||
theta2i == (theta1i+180))
    headOn = 1;
    if unknownCount > 1
        error('Head-On Collision case, too many unknowns (Max 1 unknown)')
    end
else
    if unknownCount > 2
        error('Too many unknowns, exiting out')
    elseif unknownCount < 2
        error('Overdefined problem, exiting out')
    end
end



%% ***************************************
%This part makes approximations for cases that yield division by zero
%-----------------------------------------
    if theta1i == 0 || theta1i==90 ||theta1i==180||theta1i == 270
        theta1i = theta1i + 0.00001;
    end
    if theta2i == 0 ||theta2i==90 ||theta2i==180||theta2i == 270
        theta2i = theta2i + 0.00001;
    end
    if theta1f == 0 ||theta1f==90 ||theta1f==180||theta1f == 270
        theta1f = theta1f + 0.00001;
    end
    if theta2f == 0 ||theta2f==90 ||theta2f==180||theta2f == 270
        theta2f = theta2f + 0.00001;
    end
    if theta12f == 0 ||theta12f==90 ||theta12f==180||theta12f == 270
        theta12f = theta12f + 0.00001;
    end
%End Approximations





%% ***************************************
%This part adjusts known velocities for:
% -Change in potential energy
% -Initial skid distances
%-----------------------------------------
```

```
% if(changeHeight ~= 0 || height1~=0 && height2~=0 && height12~=0 &&
heightFinal1~=0 && heightFinal~=0)
%     display('---Change in energy present; adjusting velocities to represent
values closest to impact---')
% end
height1 = str2num(get(handles.height1_o,'String'));
height2 = str2num(get(handles.height2_o,'String'));
heightFinal1 = str2num(get(handles.height1_f,'String'));
heightFinal2 = str2num(get(handles.height2_f,'String'));
height12 = str2num(get(handles.height12_f,'String'));

preSkid1 = str2num(get(handles.skid1_o,'String'));
preSkid2 = str2num(get(handles.skid2_o,'String'));
postSkid1 = str2num(get(handles.skid1_f,'String'));
postSkid2 = str2num(get(handles.skid2_f,'String'));
postSkid12 = str2num(get(handles.skid12_f,'String'));

    if v1i_ask == 1
        v1i_c = sqrt(v1i^2 - 2*g*height1 -2*u_k*g*preSkid1);
        if v1i_c <= 0
            set(handles.V_o1_c,'String','ERROR')
        else
        set(handles.V_o1_c,'String',v1i_c);
        end
    end

    if v2i_ask == 1
        v2i_c = sqrt(v2i^2 - 2*g*height2 -2*u_k*g*preSkid2);
        if v2i_c <= 0
            set(handles.V_o2_c,'String','ERROR')
        else
        set(handles.V_o2_c,'String',v2i_c);
        end
    end

    if v12f_ask == 1
        v12f_c = sqrt(v12f^2 + 2*g*height12 +2*u_k*g*postSkid12);
        if v12f_c <= 0
            set(handles.V_f12_c,'String','ERROR')
        else
        set(handles.V_f12_c,'String',v12f_c);
        end
    end

    if v1f_ask == 1
        v1f_c = sqrt(v1f^2 + 2*g*heightFinal1 +2*u_k*g*postSkid1);
        if v1f_c <= 0
            set(handles.V_f1_c,'String','ERROR')
        else
        set(handles.V_f1_c,'String',v1f_c);
        end
    end

    if v2f_ask == 1
        v2f_c = sqrt(v2f^2 + 2*g*heightFinal2 +2*u_k*g*postSkid2);
        if v2f_c <= 0
```

```
            set(handles.V_f2_c,'String','ERROR')
        else
        set(handles.V_f2_c,'String',v2f_c);
        end
    end

%*****************************************



%% **********************************************************************
%This part classifies which vehicle has known velocity and directions and
%calculates some preliminary momentum magnitudes.
%------------------------------------------------------------------------
if v1i_ask == 1 && theta1i_ask == 1
    knownVehicle = 1;
    unknownVehicle = 2;
    P1_i=m1*v1i_c;       %initial momentum magnitude of vehicle 1

elseif v2i_ask == 1 && theta2i_ask == 1
    knownVehicle = 2;
    unknownVehicle = 1;
    P2_i=m2*v2i_c;       %initial momentum magnitude of vehicle 2
end


if (stick == 1 && v12f_ask == 1)
    P12_f=m12*v12f_c;          %final momentum magnitude of vehicles 1&2
stuck together
    P1_f=0;
    P2_f=0;
    v1f_c = 0;
    v2f_c = 0;
    theta1f = 0;
    theta2f = 0;
end

if (stick == 0 && v1f_ask == 1 && v2f_ask == 1)
    P12_f = 0;
    P1_f=m1*v1f_c;
    P2_f=m2*v2f_c;
end




 %   display(['Initial velocity and direction of vehicle
',num2str(knownVehicle),' known'])
 %   display(['Solving for vehicle', num2str(unknownVehicle)])
%------------------------------------------------------------------------






%% ************************************************************************
```

```
%Case A
%This part solves for the unknown initial velocity vector of a vehicle when
%the magnitude and direction of the post-collision vehicle(s) are known,
%and the initial velocity magnitude and direction of the other vehicle
%is known.
%
%Knowns:
%-Final Direction and Magnitude of post-collision vehicles
%-Initial Direction and Magnitude of one vehicle
%--------------------------------------------------------------------------

%The following if statement checks to see if the following conditions are
%true:
%--> (final velocity vector AND (initial velocity of vehicle 1 OR 2)
%--> (final velocity vectorS AND (initial velocity of vehicle 1 OR 2)
if  (headOn == 0)&&((v12f_ask == 1 && theta12f_ask == 1 && stick == 1) ||...
    (v1f_ask == 1 && theta1f_ask == 1 && v2f_ask == 1 && theta2f_ask ==1 &&
stick == 0))&&...
    (v1i_ask==1 && theta1i_ask ==1 ||...
    v2i_ask==1 && theta2i_ask ==1)
display('Case A')
validCase = 1;

    if v1i_ask == 0
        v1i_cx = 1/m1*(m12*v12f_c*sind(theta12f)-m2*v2i_c*sind(theta2i));
        v1i_cy = 1/m1*(m12*v12f_c*cosd(theta12f)-m2*v2i_c*cosd(theta2i));
        v1i_c  = sqrt(v1i_cx^2 + v1i_cy^2);
        theta1i = rad2deg(atan2(v1i_cx,v1i_cy));
    elseif v2i_ask == 0
        v2i_cx = 1/m2*(m12*v12f_c*sind(theta12f)-m1*v1i_c*sind(theta1i));
        v2i_cy = 1/m2*(m12*v12f_c*cosd(theta12f)-m1*v1i_c*cosd(theta1i));
        v2i_c  = sqrt(v2i_cx^2 + v2i_cy^2);
        theta2i = rad2deg(atan2(v2i_cx,v2i_cy));
    end
end
%**********************************************************************


%% **********************************************************************
%Case C
%This part solves for the unknown initial velocity magnitude of a vehicle
%when the direction and magnitude of the post-collision vehicle(s) are known,
%and the initial velocity direction of both vehicles is known.
%
%Knowns:
%-Final Direction and Magnitude of post-collision vehicles
%-Initial Direction of both vehicles
%----------------------------------------------------------------------
if  ((v12f_ask == 1 && theta12f_ask == 1 && stick == 1)...
    || (stick == 0 && v1f_ask == 1 && theta1f_ask == 1 && v2f_ask == 1 &&
theta2f_ask == 1))...
    && ((v1i_ask==0 && theta1i_ask ==1 && v2i_ask==0 && theta2i_ask ==1))
display('Case C')
validCase = 1;
```

```
    v1i_c = -m12*v12f*(cosd(theta2i)*sind(theta12f)-
cosd(theta12f)*sind(theta2i))/m1/(-
cosd(theta2i)*sind(theta1i)+sind(theta2i)*cosd(theta1i));
    v2i_c = m12*v12f*(-
sind(theta1i)*cosd(theta12f)+sind(theta12f)*cosd(theta1i))/m2/(-
cosd(theta2i)*sind(theta1i)+sind(theta2i)*cosd(theta1i));
end


%% *************************************************************************
%Case B & D
%This part solves for the unknown initial velocity magnitude of a vehicle
%when the direction of the post-collision vehicle(s) are known,
%and the initial velocity direction of both vehicles is known and the
%magnitude of one initial velocity is known
%
%Knowns:
%-Final Direction of post-collision vehicles
%-Final Magnitude of one post-collision vehicle
%-Initial Magnitude & Direction of one vehicle
%-Initial Direction of both vehicles
%------------------------------------------------------------------------

if  ((((v1f_ask == 0 && v2f_ask == 1) || (v1f_ask == 1 && v2f_ask ==
0))&&(theta1f_ask==1&&theta2f_ask==1))||...
        (v12f_ask == 0 && theta12f_ask == 1 && stick == 1))...
    && ((v1i_ask==1 && theta1i_ask ==1 && v2i_ask==0 && theta2i_ask ==1)||...
        (v2i_ask==1 && theta2i_ask ==1 && v1i_ask==0 && theta1i_ask ==1))

    display('Case D')
    validCase = 1;
    if v2i_ask==0 && v2f_ask==0 && stick == 0
        A = sind(theta2i)/sind(theta2f)-cosd(theta2i)/cosd(theta2f);
        B = sind(theta2f)/sind(theta2f)-cosd(theta1f)/cosd(theta2f);
        C = sind(theta1i)/sind(theta2f)-cosd(theta1i)/cosd(theta2f);
        v2i_c = 1/(m2*A)*((m1*v1f_c)*B-m1*v1i_c*C);
        v2f_c =
1/(m2*sind(theta2f))*(m1*v1i_c*sind(theta1i)+m2*v2i_c*sind(theta2i)-
m1*v1f_c*sind(theta1f));
    end

    if v2i_ask==0 && v1f_ask==0 && stick == 0
        A = sind(theta2i)/sind(theta1f)-cosd(theta2i)/cosd(theta1f);
        B = sind(theta2f)/sind(theta1f)-cosd(theta1f)/cosd(theta1f);
        C = sind(theta1i)/sind(theta1f)-cosd(theta1i)/cosd(theta1f);
        v2i_c = 1/(m2*A)*((m2*v2f_c)*B-m1*v1i_c*C);
        v1f_c =
1/(m1*sind(theta1f))*(m1*v1i_c*sind(theta1i)+m2*v2i_c*sind(theta2i)-
m2*v2f_c*sind(theta2f));
    end

    if v1i_ask==0 && v2f_ask==0 && stick == 0
        A = sind(theta2i)/sind(theta2f)-cosd(theta2i)/cosd(theta2f);
        B = sind(theta2f)/sind(theta2f)-cosd(theta1f)/cosd(theta2f);
        C = sind(theta1i)/sind(theta2f)-cosd(theta1i)/cosd(theta2f);
        v1i_c = 1/(m1*C)*((m1*v1f_c)*B-m2*v2i_c*A);
        v2f_c =
1/(m2*sind(theta2f))*(m1*v1i_c*sind(theta1i)+m2*v2i_c*sind(theta2i)-
m1*v1f_c*sind(theta1f));
    end

    if v1i_ask==0 && v1f_ask==0 && stick == 0
        A = sind(theta2i)/sind(theta1f)-cosd(theta2i)/cosd(theta1f);
        B = sind(theta2f)/sind(theta1f)-cosd(theta1f)/cosd(theta1f);
        C = sind(theta1i)/sind(theta1f)-cosd(theta1i)/cosd(theta1f);
        v1i_c = 1/(m1*C)*((m2*v2f_c)*B-m2*v2i_c*A);
        v1f_c =
1/(m1*sind(theta1f))*(m1*v1i_c*sind(theta1i)+m2*v2i_c*sind(theta2i)-
m2*v2f_c*sind(theta2f));
    end

    if v2i_ask==0 && v12f_ask==0 && stick == 1
        A = sind(theta2i)/sind(theta12f)-cosd(theta2i)/cosd(theta12f);
        B = cosd(theta1i)/cosd(theta12f)-sind(theta1i)/sind(theta12f);
        v2i_c = 1/(m2*A)*(m1*v1i_c)*B;
        v12f_cx = 1/(m12)*(m1*v1i_c*sind(theta1i)+m2*v2i_c*sind(theta2i));
        v12f_cy = 1/(m12)*(m1*v1i_c*cosd(theta1i)+m2*v2i_c*cosd(theta2i));
        v12f_c  = sqrt(v12f_cx^2+v12f_cy^2);
    end

    if v1i_ask==0 && v12f_ask==0 && stick == 1
        A = sind(theta1i)/sind(theta12f)-cosd(theta1i)/cosd(theta12f);
        B = cosd(theta2i)/cosd(theta12f)-sind(theta2i)/sind(theta12f);
        v1i_c = 1/(m1*A)*(m2*v2i_c)*B;
        v12f_cx = 1/(m12)*(m1*v1i_c*sind(theta1i)+m2*v2i_c*sind(theta2i));
        v12f_cy = 1/(m12)*(m1*v1i_c*cosd(theta1i)+m2*v2i_c*cosd(theta2i));
        v12f_c  = sqrt(v12f_cx^2+v12f_cy^2);
    end
end
% -------------------------

%% ***********************************************************************
%Case E - Head on collision
%This part solves for the unknown initial velocity magnitude of a vehicle
%during a head-on collision. All other parameters must be known.
%
%Knowns:
%-Final Direction of vehicles
%-Final Magnitude of vehicles
%-Initial Magnitude & Direction of one vehicle

if (headOn == 1) && ((v1i_ask == 0 && v2i_ask ==1) || (v1i_ask == 1 &&
v2i_ask ==0))
    validCase = 1;
    display('Case E')
    display(theta2i)
    if v1i_ask == 0
        v1i_cx =
1/m1*(m1*v1f_c*sind(theta1f)+m2*v2f_c*sind(theta2f)+m12*sind(theta12f)-
m2*v2i_c*sind(theta2i));
```

```
        v1i_cy =
1/m1*(m1*v1f_c*cosd(theta1f)+m2*v2f_c*cosd(theta2f)+m12*cosd(theta12f)-
m2*v2i_c*cosd(theta2i));
        v1i_c = sqrt(v1i_cx^2+v1i_cy^2);
    end

    if v2i_ask == 0
        v2i_cx =
1/m2*(m1*v1f_c*sind(theta1f)+m2*v2f_c*sind(theta2f)+m12*sind(theta12f)-
m1*v1i_c*sind(theta1i));
        v2i_cy =
1/m2*(m1*v1f_c*cosd(theta1f)+m2*v2f_c*cosd(theta2f)+m12*cosd(theta12f)-
m1*v1i_c*cosd(theta1i));
        v2i_c = sqrt(v2i_cx^2+v2i_cy^2);
    end

    if((theta12f ~= theta1i) && (theta12f ~= theta2i) && stick == 1)||...
        ((theta1f ~= theta1i) && (theta1f ~= theta2i) && stick == 0)||...
        ((theta2f ~= theta1i) && (theta2f ~= theta2i) && stick == 0)
        error('Impossible Scenario for Head On Impacts')
    end

end


%% **********************************************************************
%Case F

%Knowns:
%-Final Direction of post-collision vehicles
%-Final Magnitude of post-collision vehicles
%-Initial Magnitude one vehicle
%-Initial Direction other vehicle
%---------------------------------------------------------------------
if  (v1i_ask == 0 && theta1i_ask == 1 && v2i_ask == 1 && theta2i_ask ==
0)||...
    (v1i_ask == 1 && theta1i_ask == 0 && v2i_ask == 0 && theta2i_ask == 1)
    display('Case F')
    validCase = 1; %Checks to see if it was a valid case
    loosenConv = 0;%Set variable to loosen convergence criteria
    converged = 0;      %Set variable to check convergence
    count = 0;      %Set variable to check count
    loosenFactor = 1;   %Set variable to 1 as default for convergence
multiplier

    %Final momentum in both directions are known:

Mfx=(m1*v1f_c*sind(theta1f)+m2*v2f_c*sind(theta2f)+m12*v12f_c*sind(theta12f))
;

Mfy=(m1*v1f_c*cosd(theta1f)+m2*v2f_c*cosd(theta2f)+m12*v12f_c*cosd(theta12f))
;

    %Begin iteration - guess direction, and determine resultant magnitude
    %Note that all iterations of theta are initialized at 0
    while converged ~= 1
        if v2i_ask == 0
            v2i_cy = (Mfy - m1*v1i_c*cosd(theta1i))/(m2);
            v2i_cx = (Mfx - m1*v1i_c*sind(theta1i))/(m2);
            v2i_c = sqrt(v2i_cy^2+v2i_cx^2);
        elseif v1i_ask == 0
            v1i_cy = (Mfy - m1*v2i_c*cosd(theta2i))/(m1);
            v1i_cx = (Mfx - m1*v2i_c*sind(theta2i))/(m1);
            v1i_c = sqrt(v1i_cy^2+v1i_cx^2);
        end

        %-This part checks convergence
        %Check of Conservation of Momentum
        Mix=m2*v2i_c*sind(theta2i)+m1*v1i_c*sind(theta1i);
        Miy=m2*v2i_c*cosd(theta2i)+m1*v1i_c*cosd(theta1i);

        %Convergence criteria (percent difference between initial and final
        %momentum in both directions
        convY = abs((Miy-Mfy)/Mfy);
        convX = abs((Mix-Mfx)/Mfx);

        %Check if converged
        if convY<0.0001*loosenFactor && convX<.0001*loosenFactor
            converged = 1;
        end
        %--------

        %If not converged, continue iterating
        if converged ~= 1
            if v2i_ask == 0 && v1i_ask == 1
                fprintf('theta1i = %3.5f convY = %3.6f convX = %3.6f count =
%3.0f \n',theta1i,convY,convX,count); %Print progress
                theta1i = theta1i+0.1; %Increment Theta
                count = count + 1; %Increment Count
            elseif v1i_ask == 0 && v2i_ask == 1
                fprintf('theta2i = %3.5f convY = %3.6f convX = %3.6f count =
%3.0f \n',theta2i,convY,convX,count); %Print progress
                theta2i = theta2i+0.1; %Increment Theta
                count = count + 1; %Increment Count
            end
        end

        %Relaxes convergence criteria and resets if solution didnt converge
        if count > 3600 && loosenConv == 0
            display('=========Did not converge with 0.01% criteria. Switching
to 1% criteria======')
            if v2i_ask == 0 && v1i_ask == 1
                theta1i = 0; %Reset theta back to 0
            elseif v1i_ask == 0 && v2i_ask == 1
                theta2i = 0; %Reset theta back to 0
            end
            loosenConv = 1; %Convergence Criteria relaxed (1st stage
relaxation)
            loosenFactor = 100;
            count = 0;
        end
        %Further Relaxes convergence criteria and resets if solution didnt
        %converge after 1st stage relaxation
        if count > 3600 && loosenConv == 1
            display('=========Did not converge with 1% criteria. Switching to
2% criteria======')
            if v2i_ask == 0 && v1i_ask == 1
                theta1i = 0; %Reset theta back to 0
            elseif v1i_ask == 0 && v2i_ask == 1
                theta2i = 0; %Reset theta back to 0
            end
            loosenConv = 2; %Convergence Criteria relaxed (2nd stage
relaxation)
            loosenFactor = 200;
            count = 0;
        end
        %End reset

        %Exits out if solution did not converge after 2nd stage of
        %relaxation
        if count > 3600 && loosenConv == 2
            converged = 1;
            error('Did not converge')
        end
    end

display('============')
    if loosenConv == 0
        display('Iterative solution found with 0.01% Error between initial
and final momentums')
    elseif loosenConv == 1
        display('Solution did not converge using 0.01% convergence criteria,
convergence criteria have been relaxed')
        display('Iterative solution found with 1% Error between initial and
final momentums')
    elseif loosenConv == 2
        display('Solution did not converge using 0.01% convergence criteria,
convergence criteria have been relaxed')
        display('Iterative solution found with 2% Error between initial and
final momentums')
    end
display('============')


end




%% ***********************************************************************
%Case G

%Knowns:
%-Final Direction of ONE post-collision vehicles
%-Final Magnitude of post-collision vehicles
%-Initial Magnitude one vehicle
```

```
%-Initial Direction other vehicles
%----------------------------------------------------------------------
if  ((v1i_ask == 0 && theta1i_ask == 1 && v2i_ask == 1 && theta2i_ask ==
1)||...
    (v1i_ask == 1 && theta1i_ask == 1 && v2i_ask == 0 && theta2i_ask ==
1))&&...
    (((v1f_ask == 1 && theta1f_ask == 0 && v2f_ask == 1 && theta2f_ask ==
1)||...
    (v1f_ask == 1 && theta1f_ask == 1 && v2f_ask == 1 && theta2f_ask ==
0))||...
    (v12f_ask ==1 && theta12f_ask ==0))

    display('Case G')
    validCase = 1;

    converged = 0;
    count = 0;
    loosenFactor = 1;
    loosenConv = 0;


    while converged ~= 1
        count = count + 1;

Mfx=(m1*v1f_c*sind(theta1f)+m2*v2f_c*sind(theta2f)+m12*v12f_c*sind(theta12f))
;

Mfy=(m1*v1f_c*cosd(theta1f)+m2*v2f_c*cosd(theta2f)+m12*v12f_c*cosd(theta12f))
;
        if v2i_ask == 0
            v2i_cy = (Mfy - m1*v1i_c*cosd(theta1i))/(m2);
            v2i_cx = (Mfx - m1*v1i_c*sind(theta1i))/(m2);
            v2i_c = sqrt(v2i_cy^2+v2i_cx^2);
        elseif v1i_ask == 0
            v1i_cy = (Mfy - m2*v2i_c*cosd(theta2i))/(m1);
            v1i_cx = (Mfx - m2*v2i_c*sind(theta2i))/(m1);
            v1i_c = sqrt(v1i_cy^2+v1i_cx^2);
        end

        %Check of Conservation of Momentum
        Mix=m2*v2i_c*sind(theta2i)+m1*v1i_c*sind(theta1i);
        Miy=m2*v2i_c*cosd(theta2i)+m1*v1i_c*cosd(theta1i);

Mfx=(m1*v1f_c*sind(theta1f)+m2*v2f_c*sind(theta2f)+m12*v12f_c*sind(theta12f))
;

Mfy=(m1*v1f_c*cosd(theta1f)+m2*v2f_c*cosd(theta2f)+m12*v12f_c*cosd(theta12f))
;

        convY = abs((Miy-Mfy)/Mfy);
        convX = abs((Mix-Mfx)/Mfx);



        if convY<0.0001*loosenFactor && convX<.0001*loosenFactor
```

```
            converged = 1;
        end

        if converged ~= 1
            if theta1f_ask == 0 && stick == 0
                fprintf('theta1f = %3.5f convY = %3.6f convX = %3.6f count =
%3.0f \n',theta1f,convY,convX,count);
                theta1f = theta1f+0.1;
            elseif theta2f_ask == 0 && stick == 0
                fprintf('theta2f = %3.5f convY = %3.6f convX = %3.6f count =
%3.0f \n',theta2f,convY,convX,count);
                theta2f = theta2f+0.1;
            elseif theta12f_ask == 0 && stick == 1
                fprintf('theta12f = %3.5f convY = %3.6f convX = %3.6f count =
%3.0f \n',theta12f,convY,convX,count);
                theta12f = theta12f+0.1;
            end
        end

        %Relaxes convergence criteria and resets if solution didnt converge
        if count > 3600 && loosenConv == 0
            display('=========Did not converge with 0.01% criteria. Switching
to 1% criteria======')
            if theta1f_ask == 0 && stick == 0
                theta1f = 0;
            elseif theta2f_ask == 0 && stick == 0
                theta2f = 0;
            elseif theta12f_ask == 0 && stick == 1
                theta12f = 0;
            end
            loosenConv = 1;
            loosenFactor = 100;
            count = 0;
        end

        if count > 3600 && loosenConv == 1
            display('=========Did not converge with 1% criteria. Switching to
2% criteria======')
            if theta1f_ask == 0 && stick == 0
                theta1f = 0;
            elseif theta2f_ask == 0 && stick == 0
                theta2f = 0;
            elseif theta12f_ask == 0 && stick == 1
                theta12f = 0;
            end
            loosenConv = 2;
            loosenFactor = 200;
            count = 0;
        end
        %End reset

        if count > 3600 && loosenConv == 2
            converged = 1;
            error('Did not converge')
        end
```

```
    end
    display('============')
    if loosenConv == 0
        display('Iterative solution found with 0.01% Error between initial
and final momentums')
    elseif loosenConv == 1
        display('Solution did not converge using 0.01% convergence criteria,
convergence criteria have been relaxed')
        display('Iterative solution found with 1% Error between initial and
final momentums')
    elseif loosenConv == 2
        display('Solution did not converge using 0.01% convergence criteria,
convergence criteria have been relaxed')
        display('Iterative solution found with 2% Error between initial and
final momentums')
    end
    display('============')
end



%% ******************************************************************
%This part checks to see if the case was valid
%-------------------------------------------------------------------
if validCase == 0
    error('Not a Supported Case, Exiting out. Make sure to have at least 1
initial velocity unknown')
end
%Check of Conservation of Momentum
if converged ~=1
    Mix=m1*v1i_c*sind(theta1i)+m2*v2i_c*sind(theta2i);

Mfx=m1*v1f_c*sind(theta1f)+m2*v2f_c*sind(theta2f)+m12*v12f_c*sind(theta12f);

    %Miy=(m1*v1i_c*cosd(theta1i)+m2*v2i_c*cosd(theta2i))
    Miy=(m1*v1i_c*cosd(theta1i)+m2*v2i_c*cosd(theta2i));

Mfy=(m1*v1f_c*cosd(theta1f)+m2*v2f_c*cosd(theta2f)+m12*v12f_c*cosd(theta12f))
;
end
    if abs((Mix - Mfx)/Mfx) > 0.02 || abs((Miy - Mfy)/Mfy)> 0.02
        error('Error in Case, conservation of momentum check failed. May be
an impossible scenario')
    else
        display('Final Conservation of Momentum Check Passed')
    end
%-------------------------------------------------------------------

%% ******************************************************************
%This part adjusts for final skid / changes in height, and displays the
%calculated values.
%-------------------------------------------------------------------

if v2i_ask==0
```

```
    set(handles.V_o2_c,'String',v2i_c);
    v2i = sqrt(v2i_c^2 + 2*g*height2 + 2*u_k*g*preSkid2); %Adjust for energy
change
    set(handles.V_o2,'String',v2i);
    set(handles.V_o2,'BackgroundColor','green')
end

if theta2i_ask==0
    set(handles.V_o2D,'String',theta2i);
    set(handles.V_o2D,'BackgroundColor','green')
end

if v1i_ask == 0
    set(handles.V_o1_c,'String',v1i_c);
    v1i = sqrt(v1i_c^2 + 2*g*height1 + 2*u_k*g*preSkid1);
    set(handles.V_o1,'String',v1i);
    set(handles.V_o1,'BackgroundColor','green')
end

if theta1i_ask==0
    set(handles.V_o1D,'String',theta1i);
    set(handles.V_o1D,'BackgroundColor','green')
end

if stick ==0
    if v1f_ask == 0
        set(handles.V_f1_c,'String',v1f_c);
        v1f = sqrt(v1f_c^2 - 2*g*heightFinal1 - 2*u_k*g*postSkid1);
        set(handles.V_f1,'String',v1f);
        set(handles.V_f1,'BackgroundColor','green')
    end

    if v2f_ask == 0
        set(handles.V_f2_c,'String',v2f_c);
        v2f = sqrt(v2f_c^2 - 2*g*heightFinal2 - 2*u_k*g*postSkid2);
        set(handles.V_f2,'String',v2f);
        set(handles.V_f2,'BackgroundColor','green')
    end


    if theta1f_ask==0
        set(handles.V_f1D,'String',theta1f);
        set(handles.V_f1D,'BackgroundColor','green')
    end

    if theta2f_ask==0
        set(handles.V_f2D,'String',theta2f);
        set(handles.V_f2D,'BackgroundColor','green')
    end

elseif stick == 1
    if v12f_ask == 0
        set(handles.V_f12_c,'String',v12f_c);
        v12f = sqrt(v12f_c^2 - 2*g*height12 - 2*u_k*g*postSkid12);
        set(handles.V_f12,'String',v12f);
        set(handles.V_f12,'BackgroundColor','green')
    end

    if theta12f_ask==0
        set(handles.V_f12D,'String',theta12f);
        set(handles.V_f12D,'BackgroundColor','green')
    end
end
%----------------------------------------------------------------------


%Updates all fields with changed values
guidata(hObject, handles);
%End update

%% *********************************
%This part plots the velocity vectors
%------------------------------------
if get(handles.velocityPlot,'Value')==1
    [initialVelocity1_x, initialVelocity1_y] = pol2cart(deg2rad(theta1i),
v1i_c);
    [initialVelocity2_x, initialVelocity2_y] = pol2cart(deg2rad(theta2i),
v2i_c);
    [finalVelocity12_x, finalVelocity12_y] = pol2cart(deg2rad(theta12f),
v12f_c);
    [finalVelocity1_x, finalVelocity1_y] = pol2cart(deg2rad(theta1f), v1f_c);
    [finalVelocity2_x, finalVelocity2_y] = pol2cart(deg2rad(theta2f), v2f_c);

    figure %Creates New Figure
    scale_holder = compass(1.25*max([v1i_c, v2i_c, v12f_c]), 0);
    title('Instantaneous pre/post collision velocity vectors')
    set(scale_holder, 'Visible', 'Off');
    hold on
    V1i = compass(initialVelocity1_x, initialVelocity1_y, 'b-');
    V2i = compass(initialVelocity2_x, initialVelocity2_y, 'g-');
    if stick == 1
        V12f = compass(finalVelocity12_x, finalVelocity12_y, 'r-.');
        legend([V1i, V2i, V12f], 'Vehicle 1 Initial Velocity', 'Vehicle 2
Initial Velocity', 'Vehicle12 Final Velocity', 'Location',
'NorthWestOutside')
    else
        V1f = compass(finalVelocity1_x, finalVelocity1_y, 'm:');
        V2f = compass(finalVelocity2_x, finalVelocity2_y, 'c:');
        legend([V1i, V2i, V1f, V2f], 'Vehicle 1 Initial Velocity', 'Vehicle 2
Initial Velocity', 'Vehicle1 Final Velocity','Vehicle2 Final Velocity',
'Location', 'NorthWestOutside')

    end
    view(90,-90)
end
%------------------------------------
```

Appendix H Derivation for VAR Cases

```
%Matlab Derivation for Cases C, B and D
%-----------------------------------------------------------------------
%---This part defines the variables used in the momentum equations---
syms m1 v1i theta1i v2i theta2i m2 m12 v12f theta12f
%-----------------------------------------------------------------------



%-----------------------------------------------------------------------
%---This part defines the momentum equations for 2 vehicle (stick)
%collisions---
eqn1 = 'm1*v1i*cosd(theta1i)+m2*v2i*cosd(theta2i)=m12*v12f*cosd(theta12f)';
eqn2 = 'm1*v1i*sind(theta1i)+m2*v2i*sind(theta2i)=m12*v12f*sind(theta12f)';
%-----------------------------------------------------------------------

%-----------------------------------------------------------------------
%---This part solves the momentum equations for initial velocities, in
%terms of the other stated variables
[eq1]=solve(eqn1,v1i);   %This is the equation for vli
[eq2]=solve(eqn2,v2i); %This is the equation for v2i
%-----------------------------------------------------------------------



%=======================================================================
%Case C Derivation
%-----------------------------------------------------------------------

%---This part substitutes in equation 2 into the variable "v2i", which
%yields the equation for v1i without v2i in the equation
eqn3 = subs(eq1,v2i,eq2); %This is the equation for v1i

%---This part substitutes in equation 1 into the variable "v1i", which
%yields the equation for v1i without v1iin the equation
%Case C Derivation
eqn4 = subs(eq2, v1i, eq1);%This is the equation for v2i

%The outputs of eqn3 and eqn4 yield the following in the matlab window.
%They are copy and pasted as follows for convenience:
eqn3 = '(-(-
m1*v1i*sind(theta1i)+m12*v12f*sind(theta12f))/sind(theta2i)*cosd(theta2i)+m12
*v12f*cosd(theta12f))/m1/cosd(theta1i)=v1i';
solve(eqn3,v1i) %This is the equation for v1i, without v2i as a contributing
variable
%Yields this:
%v1i=-m12*v12f*(cosd(theta2i)*sind(theta12f)-
cosd(theta12f)*sind(theta2i))/m1/(-
cosd(theta2i)*sind(theta1i)+sind(theta2i)*cosd(theta1i))

eqn4 = '(-(-
m2*v2i*cosd(theta2i)+m12*v12f*cosd(theta12f))/cosd(theta1i)*sind(theta1i)+m12
*v12f*sind(theta12f))/m2/sind(theta2i)=v2i';
%This part solves equation 4 to get v2i
```

```
solve(eqn4,v2i) %This is the equation for v2i, without v1i as a contributing
variable
%Yields this:
%%v2i=m12*v12f*(-
sind(theta1i)*cosd(theta12f)+sind(theta12f)*cosd(theta1i))/m2/(-
cosd(theta2i)*sind(theta1i)+sind(theta2i)*cosd(theta1i))
%====================================================================




%====================================================================
%Case B and D Derivation
%--------------------------------------------------------------------
%---This part solves the momentum equations for final velocities, in
%terms of the other stated variables
[eq3]=solve(eqn1,v12f); %This is the equation for v12f in the y direction
[eq4]=solve(eqn2,v12f); %This is the equation for v12f in the x direction
%--------------------------------------------------------------------

%---This part substitutes in equation 4 into the variable "v12f", which
%yields the momentum equation without v12f as a variable.
display('Case B and D')
eqn5 = subs(eq1,v12f,eq4); %This is the equation for v1i without v12f in the
equation
eqn6 = subs(eq2,v12f,eq3); %This is the equation for v2i without v12f in the
equation

%The outputs of eqn5 and eqn6 yield the following in the matlab window.
%They are copy and pasted as follows for convenience:
%eqn5 =(-
m2*v2i*cosd(theta2i)+(m1*v1i*sind(theta1i)+m2*v2i*sind(theta2i))/sind(theta12
f)*cosd(theta12f))/m1/cosd(theta1i)
%eqn6
%=(-
m1*v1i*sind(theta1i)+(m1*v1i*cosd(theta1i)+m2*v2i*cosd(theta2i))/cosd(theta12
f)*sind(theta12f))/m2/sind(theta2i)
eqn5 = '(-
m2*v2i*cosd(theta2i)+(m1*v1i*sind(theta1i)+m2*v2i*sind(theta2i))/sind(theta12
f)*cosd(theta12f))/m1/cosd(theta1i) = v1i';
eqn6 = '-(m1*v1i*sind(theta1i)-
(m1*v1i*cosd(theta1i)+m2*v2i*cosd(theta2i))/cosd(theta12f)*sind(theta12f))/m2
/sind(theta2i) = v2i';

solve(eqn5,v1i)
%Yields this:
%v1i = m2*v2i*(-
cosd(theta2i)*sind(theta12f)+cosd(theta12f)*sind(theta2i))/m1/(-
sind(theta1i)*cosd(theta12f)+sind(theta12f)*cosd(theta1i))

solve(eqn6,v2i)
%v2i = m1*v1i*(-
sind(theta1i)*cosd(theta12f)+sind(theta12f)*cosd(theta1i))/m2/(-
cosd(theta2i)*sind(theta12f)+cosd(theta12f)*sind(theta2i))
%====================================================================
```

Appendix I Catapult Postprocessing Code

```
%Catapult PostProcessing Code
%Lawrence Fong
%----------------------------------------------------
%Notes:
%----------------------------------------------------
%In matrices A, AA, AA_scaled, AAA:
%(:,1) is the time
%(:,2) is the angular acceleration or tangential acceleration
%(:,3) is the angular velocity or angular acceleration
%(:,4) is the axial strain
%(:,5) is the bending strain
%A is the loaded matrix from LabView
%----------------------------------------------------
%In matrix omega:
%(:,1) is the time
%(:,2) is the angular velocity
%----------------------------------------------------
%In matrix theta:
%(:,1) is the time
%(:,2) is the angular velocity
%(:,3) is the angular position
%----------------------------------------------------
%--
%AA is the rescaled matrix, correcting for labview scaling / offsets
%--
%AA_scaled is the matrix for free catapult arm motion, with tangential and
normal accelerations in g's
%--
%AAA is the matrix for free catapult arm motion, with angular acceleration
%and angular velocity in rad/s, and rad/s^2, respectively
%--
%impactG is the matrix for impact, with acceleration in "g's"
%--
%imapact is the matrix for impact, with acceleration of rad/s^2 and velocity
%of rad/s
%--
%omega is the matrix that attempts to correct for negative velocities due
%to noise
%--
%theta is the matrix that is used to find position
%----------------------------------------------------
close all
clear all
clc
fileName=input('Please enter in the file name: ','s');
A = load(fileName);


gravOffset=0;

%Catapult Values
r = 0.336; %meters // length from pivot to accelerometer
g = 9.81;   %m/s^2 //gravity (metric)
```

```
L_stopper=3/12; %length from stopper pin to pivot in feet

%Nominal Values of Accelerometer
V_noLoad=2.5; %V when no exication
sens = 38;      %Sensitivity of accel

%Nominal Values of Strain Gage
invertStrainAxial = 1; %-1 inverts voltage from axial strain gage
invertStrainMoment = 1; %-1 inverts voltage from moment strain gage use
V_strainNoLoad = 100; %V when no load on strain gage
V_ex=2.5;             %Excitation Voltage
Sg = 2.075;           %Strain gage 'gage factor'
Rg = 120;             %Ohms, strain gage resistance

matrixSize = size(A);    %Finds size of output matrix
nn=matrixSize(1,1);       %Number of Rows



%Scaling Output to "no load voltage"
for n=1:nn
    AA(n,1)=(A(n,1)-A(1,1))*0.001;   %Rescale to time = 0 at beginning
    AA(n,2)=-(V_noLoad - A(n,2)); %Scaling to noLoad
    AA(n,3)=V_noLoad - A(n,3); %Scaling to noLoad
    AA(n,4)=invertStrainAxial*A(n,4)/(1000000); %Adjust for voltage, which
was multiplied by 1000000 in LabView, and is in
millivolts_strain/volts_exication
    AA(n,5)=invertStrainMoment*A(n,5)/(1000000); %Adjust for voltage, which
was multiplied by 1000000 in LabView, and is in
millivolts_strain/volts_exication
end

%======================================
%This part finds where the beginning and ending cutoff points should be
triggerValueNormal=0.03;  %Volts of normal acceleration
triggerValueTangential=0.1; %Volts of angular acceleration
startOffset = 20; %Samples
for n=startOffset:nn
    if(AA(n,2)>triggerValueTangential && AA(n,3)>triggerValueNormal);
        timeBeginRow=n-5;
        break %exit out of for loop
    end
end

%Finding end of free rotation of catapult arm
for n=timeBeginRow:nn
    if(AA(n,2)<-.5); %-.5 simply a experimentally determined "good" trigger
value
        timeEndRow=n-1;
        break %exit out of for loop
    end
end

%Find end of impact
for n=timeEndRow+3:nn
```

```
    if(abs(AA(n,5))<1.2*abs(AA(1,5))) %1.2 relaxes criteria for ending impact
        impactEndRow=n;
        break
    end
end
%=======================================



%=======================================
%This part cuts off irrelevant pre and post collision data, and multiplies
%it by the accelerometer scaling factor, AA_scaled results in normal and
%tangential accelerations in values of "g's"

for n=1:(timeEndRow-timeBeginRow)
    AA_scaled(n,1)=(AA(n+timeBeginRow,1)-AA(timeBeginRow,1));    %Scaling
time to zero
    AA_scaled(n,2)=(1000/sens)*(AA(n+timeBeginRow,2));    %Scaling Accel 1
    AA_scaled(n,3)=(1000/sens)*(AA(n+timeBeginRow,3)+gravOffset);    %Scaling
Accel 2
    AA_scaled(n,4)=(2/Sg)*(AA(n+timeBeginRow,4))*(1/1000); %multiply by gage
factor equation to obtain strain, multiply by 1/1000 to get V/V
    AA_scaled(n,5)=(2/Sg)*(AA(n+timeBeginRow,5)-AA(timeBeginRow,5))*(1/1000);
%multiply by gage factor equation to obtain strain, multiply by 1/1000 to get
V/V
end

%This part takes the scaled accelerometer values and converts them to
%angular velocity and acceleration
for n=1:(timeEndRow-timeBeginRow)
    AAA(n,1)=AA_scaled(n,1);    %Time Doesnt Change
    AAA(n,2)=AA_scaled(n,2)*g/r;    %Turning Tangential Accel into angular
Accel
    AAA(n,3)=sqrt(AA_scaled(n,3)*g/r);    %Turning Normal Accel into angular
Velocity
    AAA(n,4)=AA_scaled(n,4);
    AAA(n,5)=AA_scaled(n,5);
end
%======================================



%======================================
%This part captures the data for the IMPACT with the stopper pin, and
multiplies
%it by the accelerometer scaling factor.
for n=1:(impactEndRow-timeEndRow)
    impactG(n,1)=(AA(n+timeEndRow,1)-AA(timeEndRow,1));    %Scaling time to
zero
    impactG(n,2)=(1000/sens)*(AA(n+timeEndRow,2));    %Scaling Accel 1
    impactG(n,3)=(1000/sens)*(AA(n+timeEndRow,3)+gravOffset);    %Scaling
Accel 2
    impactG(n,4)=(2/Sg)*(1/1000)*(AA(n+timeEndRow,4)-AA(timeEndRow,4));
    impactG(n,5)=(2/Sg)*(1/1000)*(AA(n+timeEndRow,5)-AA(timeEndRow,5));
end
```

```
%This part takes the  accelerometer values and converts them to
%angular velocity and acceleration
for n=1:(impactEndRow-timeEndRow)
    impact(n,1)=impactG(n,1);    %Time Doesnt Change
    impact(n,2)=impactG(n,2)*g/r;    %Turning Tangential Accel into angular
Accel
    impact(n,3)=sqrt(impactG(n,3)*g/r);    %Turning Normal Accel into angular
Velocity
    impact(n,4)=impactG(n,4);
    impact(n,5)=impactG(n,5);
end
%=======================================




%=====================================
%This part takes the angular velocity, and sets the startpoint of nonzero
%velocity to time = 0
%This part finds where the the velocity is nonzero
for n=1:nn
    if(real(AAA(n,3))>0);
        timeBeginRowOmega=n-1;
        break %exit out of for loop
    end
end
%This part rezeroes the matrix to the start of nonzero velocity
sizeAAA=size(AAA); %Find size of Matrix
for n=1:(sizeAAA(1,1)-timeBeginRowOmega)
    omega(n,1)=AAA(n,1);    %Time Doesnt Change
    omega(n,2)=AAA((timeBeginRowOmega+n),3);    %Scaling Omega
end
%=====================================




%=====================================
%This part performs a rough numerical integration on the angular velocity
%in order to find position
sizeOmega=size(omega); %Find size of velocity matrix
for n=2:sizeOmega
    theta(1,1)=0;   %Initial Condition for time (miliseconds)
    theta(n,1)=omega(n-1,1);
%    theta(1,2)=deg2rad(pullBack);    %Initial Condition for position
    theta(1,2)=0;    %Initial Condition for position
    theta(n,2)=(omega(n,2)+0.5*(omega(n,2)-omega(n-1,2)))*(omega(n,1)-
omega(n-1,1))+theta(n-1,2); %(velocity*change in time)
end
thetaTraveled = rad2deg(max(theta(:,2)))


%---------------------------------------------------------
%Import Theoretical Results
[t_theo, theta_theo, omega_rod_theo, strain_ax, I_total, I_strain,
alpha_theo, M_theo, strain_moment] = catapultTheoreticalR2(thetaTraveled);
%---------------------------------------------------------.
```

```
%======================================
%This part calculates the force on the stopper pin using the impact data
array
timeImpact = max(impact(:,1)); %Find time of impact
omega_pre_impact = max(AAA(:,3)); %Angular velocity pre-impact
F_impact = 2*(omega_pre_impact)*I_total/(L_stopper*timeImpact)
%======================================



%==============================================================
figure (1)
%Plots accelerations and angular velocity
%--------------------------------------------------------------
subplot(3,1,1),plot(AA(:,1),AA(:,2),AA(:,1),AA(:,3))
xlabel('Time, s', 'fontsize', 12, 'fontweight', 'bold')
ylabel('Magnitude, Volts', 'fontweight', 'bold')
legend('Tangential Acceleration','Normal
Acceleration','Location','NorthEast')

% subplot(4,1,2),plot(AA_scaled(:,1),AA_scaled(:,4))
% xlabel('Time, s', 'fontsize', 12, 'fontweight', 'bold')
% ylabel('Magnitude, g', 'fontsize', 12, 'fontweight', 'bold')
% legend('Tangential Acceleration','Normal
Acceleration','Location','NorthWest')

subplot(3,1,2),plot(AAA(:,1),AAA(:,3),'o',t_theo,omega_rod_theo)
xlabel('Rescaled Time, s', 'fontsize', 12, 'fontweight', 'bold')
ylabel('Angular Velocity (rad/s)',  'fontweight', 'bold')
legend('Experimental Angular Velocity','Theoretical Angular
Velocity','Location','NorthWest')

subplot(3,1,3),plot(AAA(:,1),AAA(:,2),'o',t_theo,alpha_theo)
xlabel('Rescaled Time, s', 'fontsize', 12, 'fontweight', 'bold')
ylabel('Angular Acceleration (rad/s^2)',  'fontweight', 'bold')
legend('Experimental Angular Acceleration','Theoretical Angular Angular
Acceleration','Location','NorthEast')
%--------------------------------------------------------------




%==============================================================
figure (2)
%Plots strain data
%--------------------------------------------------------------
subplot(3,1,1),plot(AA(:,1),AA(:,4),AA(:,1),AA(:,5))
xlabel('Time, s', 'fontsize', 12, 'fontweight', 'bold')
legend('Axial Strain Voltage','Moment Strain Voltage','Location','NorthEast')
ylabel('Strain Voltage, mV/V', 'color', 'black', 'fontsize', 12,
'fontweight', 'bold')

subplot(3,1,2),plotyy(AA_scaled(:,1),AA_scaled(:,4),t_theo,strain_ax)
```

```
legend('Theoretical Axial Strain','Experimental Axial
Strain','Location','NorthWest')
xlabel('Rescaled Time, s', 'fontsize', 12, 'fontweight', 'bold')
ylabel('Strain ft/ft', 'color', 'black', 'fontsize', 12, 'fontweight',
'bold')

subplot(3,1,3),plotyy(AA_scaled(:,1),AA_scaled(:,5),t_theo,strain_moment)
legend('Theoretical Axial Strain','Experimental Axial
Strain','Location','NorthWest')
xlabel('Rescaled Time, s', 'fontsize', 12, 'fontweight', 'bold')
ylabel('Strain ft/ft', 'color', 'black', 'fontsize', 12, 'fontweight',
'bold')
%-------------------------------------------------------------



%=============================================================
figure (3)
%plots impact data
%-------------------------------------------------------------
subplot(4,1,1), plot(impact(:,1),impact(:,2),'o',impact(:,1),impact(:,2))
xlabel('Impact Time, s', 'fontsize', 12, 'fontweight', 'bold')
legend('Angular Acceleration','Location','NorthEast')
ylabel('(rad/s^2)',  'fontweight', 'bold')

subplot(4,1,2), plot(impact(:,1),impact(:,3),'o',impact(:,1),impact(:,3))
xlabel('Impact Time, s', 'fontsize', 12, 'fontweight', 'bold')
legend('Angular Velocity Magnitude','Location','NorthEast')
ylabel('(rad/s)',  'fontweight', 'bold')

subplot(4,1,3), plot(impact(:,1),impact(:,4),'o',impact(:,1),impact(:,4))
xlabel('Impact Time, s', 'fontsize', 12, 'fontweight', 'bold')
legend('Axial Strain','Location','NorthEast')
ylabel('Strain (in/in)','fontweight', 'bold')

subplot(4,1,4), plot(impact(:,1),impact(:,5),'o',impact(:,1),impact(:,5))
xlabel('Impact Time, s', 'fontsize', 12, 'fontweight', 'bold')
legend('Moment Strain','Location','NorthEast')
ylabel('Strain (in/in)','fontweight', 'bold')
%-------------------------------------------------------------


%=============================================================
figure (4)
%plots raw readings
%-------------------------------------------------------------
subplot(4,1,1), plot(AA(:,1),AA(:,2))
xlabel('Total Time, s', 'fontsize', 12, 'fontweight', 'bold')
legend('Tangential Acceleration Voltage','Location','NorthEast')
ylabel('Volts',  'fontweight', 'bold')

subplot(4,1,2), plot(AA(:,1),AA(:,3))
xlabel('Total Time, s', 'fontsize', 12, 'fontweight', 'bold')
legend('Normal Acceleration Voltage','Location','NorthEast')
ylabel('Volts',  'fontweight', 'bold')
```

```
subplot(4,1,3), plot(AA(:,1),AA(:,4))
xlabel('Total Time, s', 'fontsize', 12, 'fontweight', 'bold')
legend('Axial Strain Voltage','Location','NorthEast')
ylabel('mV/V', 'fontweight', 'bold')

subplot(4,1,4), plot(AA(:,1),AA(:,5))
xlabel('Total Time, s', 'fontsize', 12, 'fontweight', 'bold')
legend('Moment Strain Voltage','Location','NorthEast')
ylabel('mV/V', 'fontweight', 'bold')
%------------------------------------------------------------
```

Appendix J   Catapult Theoretical Code

```
%Theoretical Catapult Code
%Lawrence Fong
%ME599

function [t_theo, theta_theo, omega_rod_theo, strain_ax, I_total, I_strain,
alpha_theo, M_theo, strain_moment] = catapultTheoreticalR2(thetaTraveled)



%--------------------
%Catapult Parameters
%--------------------
theta_stopper = 125; %Location of stopper pin

travel_accept=input(['Accept Numerically Integrated Travel Distance?'
'(1=yes,0=no):  ']);
if travel_accept == 1
    thetaTraveled = round(thetaTraveled);
elseif travel_accept == 0
    thetaTraveled = input('Enter in angle between stopper pin and pullback
angle:  ');
else
    error('Incorrect Entry (1 or 0)')
end




thetaTraveled = round(thetaTraveled);     %80 degrees corresponds to the
100deg position on the catapult markings



g       = 32.2;       %Gravity in ft/s^2
qtyBands= 1;          %Number of Rubber Bands Used
m_arm   = 0.281/g;    %Pounds
m_cup   = 0.055/g; %Mass of the ammo cup (lbs)
m_egg   = 0.00/g;     %Mass of the egg (lbs)
m_total = m_arm+m_cup+m_egg; %Total Mass
w_arm   = (1/12);     %Dimension of arm cross section, parallel to direction
of rotation
t_arm   = (0.75/12);     %Dimension of arm cross section, perpendicular to
direction of rotation
A_arm   = (w_arm)*(t_arm); %Cross sectional area of catapult arm

L_strain= 4.75/12;     %Length from catapult arm pivot to mounted strain gages
L_arm   = 14.25/12;   %Length of catapult arm (feet)
L_u     = 15/12;  %Unstretched length of rubber band
AP      = 8.375/12;   %Length from Rubber band attachment on base to rotating
pin.
OC      = 13.25/12;   %Length from the pivot point to the ammo cup
O_x     = 6.5/12;     %X-Length from arm pivot to rubber band attachment on
base
O_y     = 0;          %Y-Length from arm pivot to base
P       = 0;          %Y-Length from base to rubber band attachment on base
```

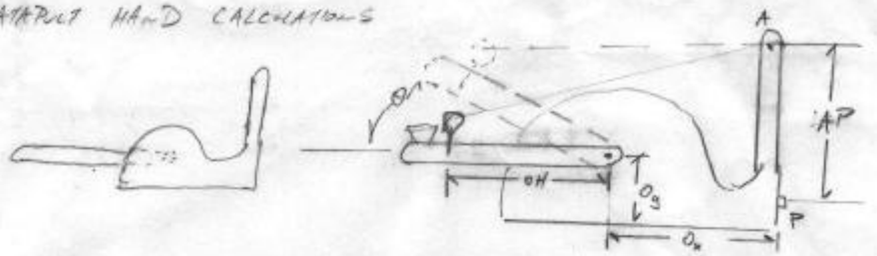
```
d_arm   = m_arm/(L_arm*t_arm*w_arm); %density of arm

E_wood  = 1600000;   %Elastic Modulus of Wood in lb/in^2

OH      = 11.75/12;  %Length from pivot pin to eye hook
thetaInit  = 180-(theta_stopper+thetaTraveled);      %0 degrees corresponds
to the 180deg position on the catapult markings
thetaFinal = thetaTraveled + thetaInit;
thetaStep  = 0.01;   %Unitless
%--------------------



%--------------------------------
% Creating Array of theta Values
%--------------------------------
theta_theo=thetaInit:thetaStep:thetaFinal;
%--------------------------------



%-----------------------------------------------------
% Finding Effective Center of Mass of Arm + Cup + Egg
%-----------------------------------------------------
L_cm=(m_arm*(L_arm/2)+(m_cup+m_egg)*OC)/(m_arm+m_cup+m_egg); %Location of
catapult arm+cup+egg center of mass
m_arm_eff = d_arm*(L_arm-L_strain)*w_arm*t_arm; %Effective arm mass above
mounted gage
m_tot_eff = m_arm_eff + m_cup + m_egg; %total mass above mounted gage
L_cm_strain = (m_arm_eff*((L_arm-
L_strain)/2+L_strain)+(m_cup+m_egg)*OC)/(m_arm_eff+m_cup+m_egg);
r_strain = (L_cm_strain - L_strain); %Distance between strain gage and center
of mass
%-----------------------------------------------------




%-------------------------------------------------
% Calculating Rubber Band Length and Displacement
%-------------------------------------------------
AH_y = (AP + P) - (OH.*sind(theta_theo) + O_y); %x-distance in feet
AH_x = O_x + OH.*cosd(theta_theo); %y-distance in feet
AH=sqrt(AH_x.^2+AH_y.^2); %Distance from rubber band attachment on arm to
pivot pin

theta_band=(atan(AH_y./AH_x)+deg2rad(theta_theo))';

%Displacement of Rubber band:
L_s = AH + AP; %Stretched length of rubber band
dL = L_s - L_u; %Displacement of rubber band (stretch - unstretch)

%Spring Constant of Rubber Band
E_band=zeros((thetaFinal-thetaInit)/thetaStep+1,1);  %Creates zero matrix to
perform for loop
E_pot=zeros((thetaFinal-thetaInit)/thetaStep+1,1);  %Creates zero matrix to
perform for loop
```

```
F_band=zeros((thetaFinal-thetaInit)/thetaStep+1,1); %Creates zero matrix to
perform for loop

%Creates an array of Energy stored in rubber band and magnitude of Force of
the rubber band for every theta position
for n=1:(thetaFinal-thetaInit)/thetaStep+1
     %Taken from experimentally determined equation trendline fit
     F_band(n)= qtyBands*(3.465*dL(n)^3 - 9.677*dL(n)^2 + 13.55*dL(n) +
0.091);
     E_band(n)= qtyBands*(3.465/4*dL(n)^4 - 9.677/3*dL(n)^3 + 13.55/2*dL(n)^2
+ 0.091*dL(n));
    E_pot(n) = m_total*L_cm*sind(theta_theo(n));
end
%------------------------------------------------



%-----------------------------------------------------
% Applying Energy Principles to find Angular Velocity
%-----------------------------------------------------
I_total = (1/3)*m_arm*(L_arm)^2 + (m_cup + m_egg)*(OC)^2; %Inertia of the
(arm + cup + egg) in ft^2
I_strain = (1/12)*m_tot_eff*(L_arm-L_strain)^2+m_tot_eff*(L_cm_strain-
r_strain)^2;%in ft^2

omega_rod_theo=zeros((thetaFinal-thetaInit)/thetaStep+1,1);
for n=1:(thetaFinal-thetaInit)/thetaStep+1
%The following equation takes the difference in starting ruber band
%energy E_band(1) and rubber band energy at E_band(n) and uses this to
%calculate the angular velocity of the system.
    omega_rod_theo(n)=sqrt(2*(E_band(1)-E_band(n)+E_pot(1)-
E_pot(n))/I_total); %angular velocity of (arm + cup + egg)
end
%-----------------------------------------------------

%----------------
% Estimating Time
%----------------
t_theo=zeros((thetaFinal-thetaInit)/thetaStep+1,1); %Creates zero matrix to
perform for loop
for n=2:(thetaFinal-thetaInit)/thetaStep+1
    t_theo(n)= t_theo(n-1) + deg2rad(theta_theo(n)-theta_theo(n-
1))/omega_rod_theo(n); %Takes theta/omega to estimate time
end
%----------------



%-----------------------------------------------
% Using Angular Velocity to Calculate for Strain
%-----------------------------------------------
%Find Normal Force:
F_n=m_tot_eff.*omega_rod_theo.^2.*r_strain;    %Finds normal force due to the
centripetal acceleration of catapult arm
F_net=F_n-F_band.*cos(theta_band);  %Finds net axial force (F_normal -
F_band_axial) on catapult arm
%Find Axial Strain
```

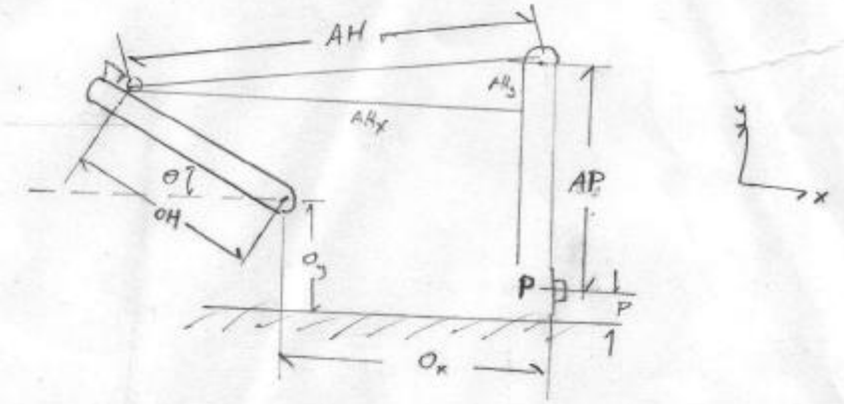
```
stress_ax = F_net./A_arm;            %Finds axial stress (lb/ft^2)
strain_ax = stress_ax./(E_wood*12^2);%Finds axial strain, E_wood is given in
inches



%Find Angular Acceleration
alpha_theo = F_band.*sin(theta_band)*OH/I_total;

%Find Theoretical Moment
M_theo = I_strain.*alpha_theo-F_band.*sin(theta_band)*(OH-
r_strain)+m_tot_eff.*(alpha_theo.*(L_cm_strain-r_strain).*(L_cm_strain-
r_strain));
%Find Theoretical Strain:
stress_moment = M_theo*(w_arm/2)/(1/12*t_arm*w_arm^3);
strain_moment = stress_moment/(E_wood*12^2);
%-----------------------------------------------
```

Appendix K Selected Catapult Hand Calculations

CATAPULT HAND CALCULATIONS



<u>FIRST</u>: NEED RUBBER BAND DISPLACEMENT AS FUNCTION OF θ



INTERMEDIATE DIMENSIONS: AH, AH_x, AH_y

$$AH_y = (AP + P) - (OH \sin\theta + O_y)$$

$$AH_x = (OH\cos\theta) + O_x$$

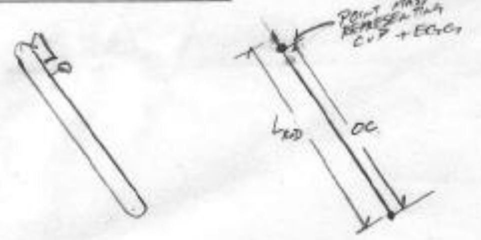
$$AH = \sqrt{AH_x^2 + AH_y^2}$$

DISPLACEMENT OF RUBBER BAND:

TOTAL LENGTH OF BAND: $L_{\text{STRETCH}} = AH + AP$

DISPLACEMENT = $dL = L_{\text{STRETCH}} - L_{\text{UNSTRETCH}}$

APPLYING ENERGY



SLENDER ROD: $I_0 = \frac{1}{3} m L^2$

POINT MASS + md^2

$$I_{TOTAL} = \frac{1}{3} m_{ARM} L_{ROD}^2 + (m_{CUP} + m_{EGG})(OC)^2$$

APPLYING ENERGY:

EXPERIMENTALLY DETERMINED RUBBER BAND CURVE:

$$F_{BAND} = 3.465(dL)^3 - 9.677(dL)^2 + 13.53(dL) + 0.091$$

INTEGRATING

$$\rightarrow E_{-BAND} = \frac{3.465}{4}(dL)^4 - \frac{9.677}{3}(dL)^3 + \frac{13.55}{2}(dL)^2 + 0.091(dL)$$

$$PE_0 + KE_0 = PE_1 + KE_1$$

$$KE_1 = (PE_0 - PE_1) + \cancel{KE_0}^{\,0}$$

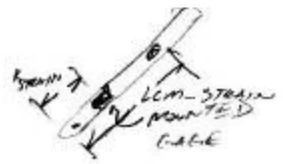
$$KE_1 = PE_0 - PE_1$$

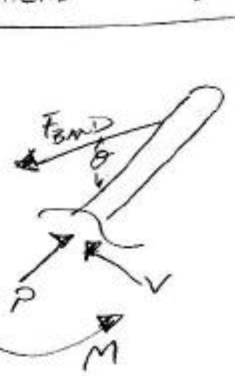
$$\frac{1}{2} I \omega^2 = E_{BAND-0} - E_{BAND-1}$$

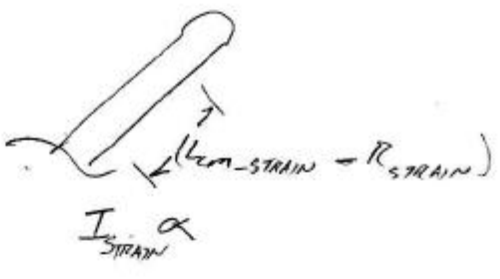
$$\omega_1 = \sqrt{\frac{2(E_{BAND-0} - E_{BAND-1})}{I_{TOTAL}}}$$

$$\boxed{\omega(n) = \sqrt{\frac{2(E_{BAND-0} - E_{BAND(n)})}{I_{TOTAL}}}}$$

THEORETICAL MOMENT



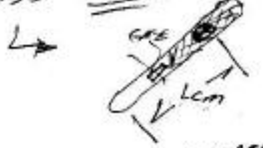




$$\Sigma M = I\alpha$$

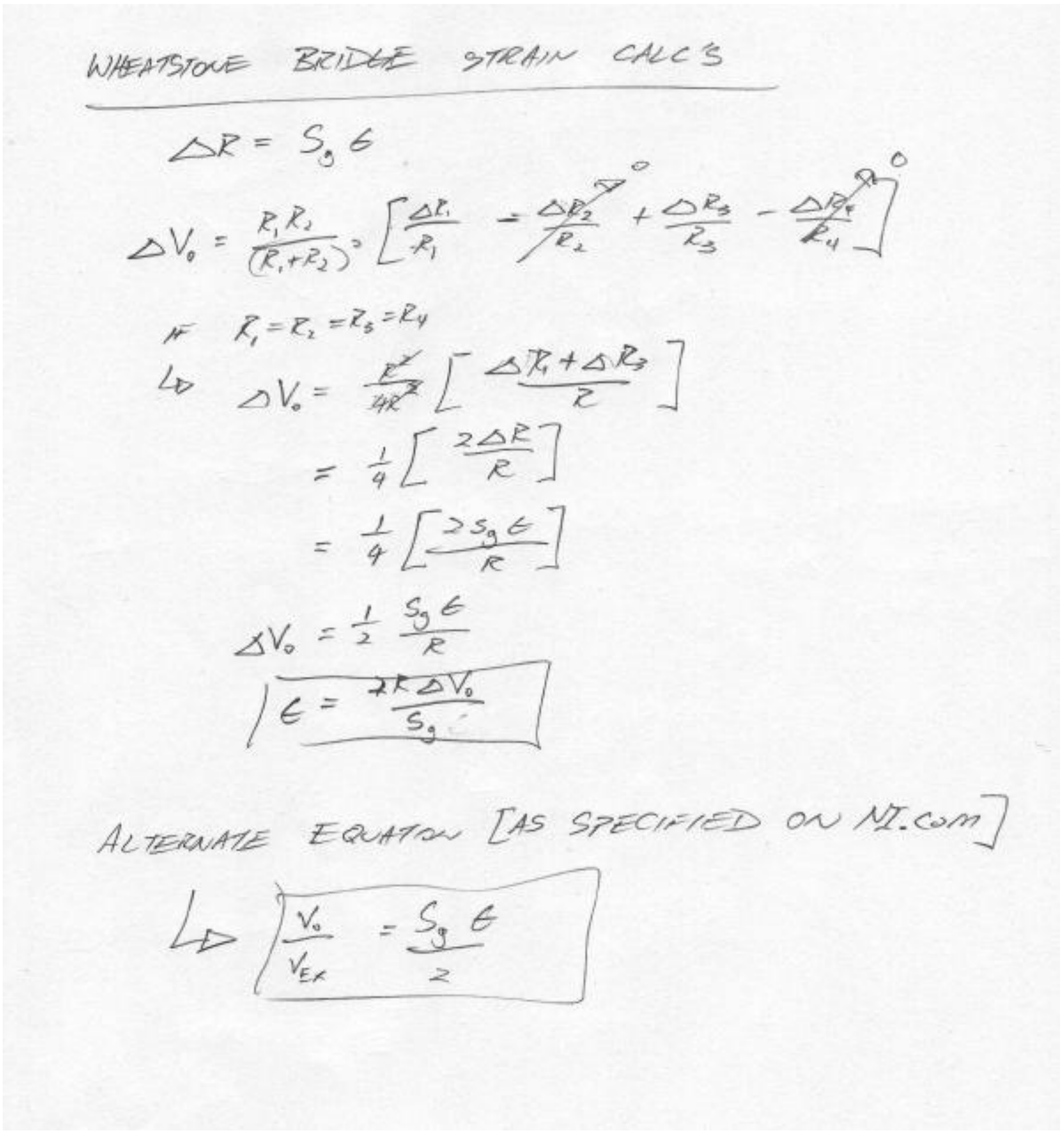
$$-M + (F\_BAND \cdot \sin\theta)(OH - R\_STRAIN) = I\_STRAIN\,\alpha + m_{EFF}\,\alpha\,(L\_cm\_STRAIN - R\_STRAIN)^2$$

L_cm_STRAIN = LOCATION OF CENTER OF MASS OF MASS ABOVE STRAIN GAGES



$$STRESS\_MOMENT = \frac{Mc}{I} = \frac{M \cdot \left(\frac{W\_ARM}{2}\right)}{\left(\frac{1}{2}bh^3\right)}$$

$$STRAIN = \frac{\sigma}{E} = \frac{STRESS}{E}$$



WHEATSTONE BRIDGE STRAIN CALC'S

$$\Delta R = S_g \epsilon$$

$$\Delta V_0 = \frac{R_1 R_2}{(R_1+R_2)^2}\left[\frac{\Delta R_1}{R_1} - \overset{0}{\cancel{\frac{\Delta R_2}{R_2}}} + \frac{\Delta R_3}{R_3} - \overset{0}{\cancel{\frac{\Delta R_4}{R_4}}}\right]$$

IF $R_1 = R_2 = R_3 = R_4$

$$\to \Delta V_0 = \frac{\cancel{R^2}}{4\cancel{R^2}}\left[\frac{\Delta R_1 + \Delta R_3}{R}\right]$$

$$= \frac{1}{4}\left[\frac{2\Delta R}{R}\right]$$

$$= \frac{1}{4}\left[\frac{2 S_g \epsilon}{R}\right]$$

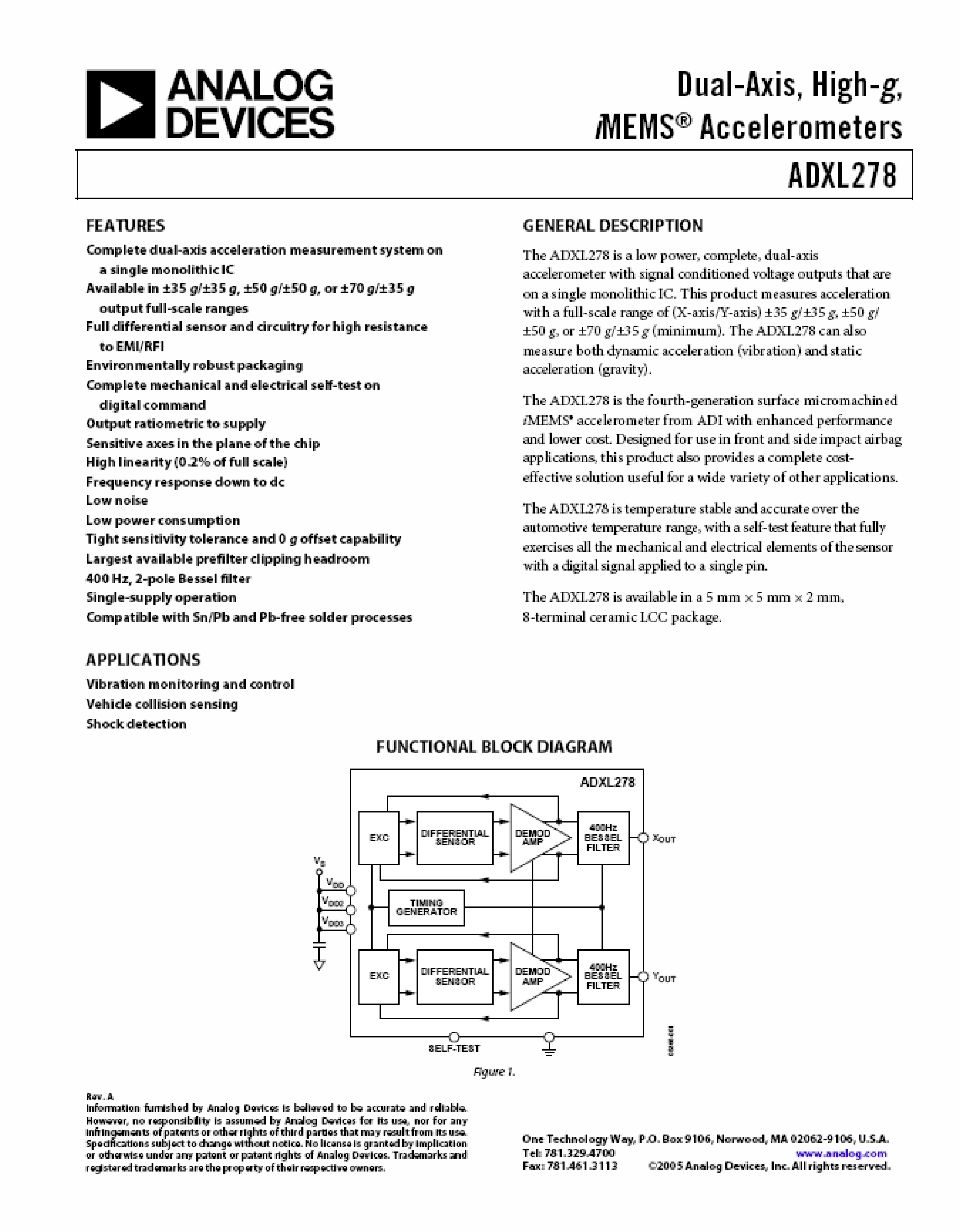
$$\Delta V_0 = \frac{1}{2}\frac{S_g \epsilon}{R}$$

$$\boxed{\epsilon = \frac{2R\,\Delta V_0}{S_g}}$$

ALTERNATE EQUATION [AS SPECIFIED ON NI.COM]

$$\to \boxed{\frac{V_0}{V_{EX}} = \frac{S_g \epsilon}{2}}$$

Appendix L ADXL-278 Specifications



ANALOG DEVICES

Dual-Axis, High-*g*, *i*MEMS® Accelerometers

ADXL278

FEATURES

Complete dual-axis acceleration measurement system on a single monolithic IC
Available in ±35 *g*/±35 *g*, ±50 *g*/±50 *g*, or ±70 *g*/±35 *g* output full-scale ranges
Full differential sensor and circuitry for high resistance to EMI/RFI
Environmentally robust packaging
Complete mechanical and electrical self-test on digital command
Output ratiometric to supply
Sensitive axes in the plane of the chip
High linearity (0.2% of full scale)
Frequency response down to dc
Low noise
Low power consumption
Tight sensitivity tolerance and 0 *g* offset capability
Largest available prefilter clipping headroom
400 Hz, 2-pole Bessel filter
Single-supply operation
Compatible with Sn/Pb and Pb-free solder processes

APPLICATIONS

Vibration monitoring and control
Vehicle collision sensing
Shock detection

GENERAL DESCRIPTION

The ADXL278 is a low power, complete, dual-axis accelerometer with signal conditioned voltage outputs that are on a single monolithic IC. This product measures acceleration with a full-scale range of (X-axis/Y-axis) ±35 *g*/±35 *g*, ±50 *g*/±50 *g*, or ±70 *g*/±35 *g* (minimum). The ADXL278 can also measure both dynamic acceleration (vibration) and static acceleration (gravity).

The ADXL278 is the fourth-generation surface micromachined *i*MEMS® accelerometer from ADI with enhanced performance and lower cost. Designed for use in front and side impact airbag applications, this product also provides a complete cost-effective solution useful for a wide variety of other applications.

The ADXL278 is temperature stable and accurate over the automotive temperature range, with a self-test feature that fully exercises all the mechanical and electrical elements of the sensor with a digital signal applied to a single pin.

The ADXL278 is available in a 5 mm × 5 mm × 2 mm, 8-terminal ceramic LCC package.

FUNCTIONAL BLOCK DIAGRAM

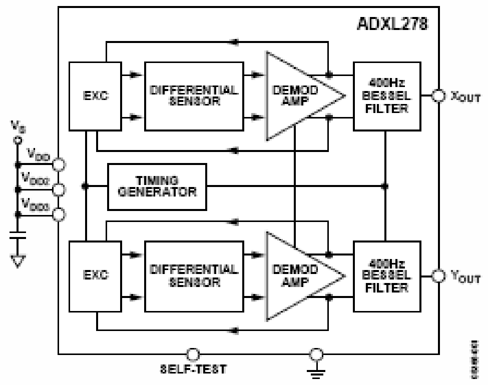


Figure 1.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 **www.analog.com**
Fax: 781.461.3113 **©2005 Analog Devices, Inc. All rights reserved.**

TABLE OF CONTENTS

REVISION HISTORY

5/05—Rev. 0 to Rev. A

SPECIFICATIONS[1]

At T_A = −40°C to +105°C, 5.0 V dc ± 5%, acceleration = 0 *g*, unless otherwise noted.

Table 1.

		Model No. AD22284			Model No. AD22285			Model No. AD22286				
Parameter	**Conditions**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Axis**	**Min**	**Typ**	**Max**	**Unit**
SENSOR												
Output Full-Scale Range	$I_{OUT} \le \pm 100\ \mu A$	37			55			X	70			*g*
								Y	37			*g*
Nonlinearity			0.2	2		0.2	2			0.2	2	%
Package Alignment Error			1			1				1		Degree
Sensor-to-Sensor Alignment Error			0.1			0.1				0.1		Degree
Cross-Axis Sensitivity		−5		+5	−5		+5		−5		+5	%
Resonant Frequency			24			24				24		kHz
Sensitivity, Ratiometric (Over Temperature)	V_{DD} = 5 V, 100 Hz	52.25	55	57.75	36.1	38	39.9	X	25.65	27	28.35	mV/*g*
								Y	52.25	55	57.75	mV/*g*
OFFSET												
Zero-*g* Output Voltage (Over Temperature)[2]	$V_{OUT} - V_{DD}/2$, V_{DD} = 5 V	−150		+150	−150		+150	X	−100		+100	mV
								Y	−150		+150	mV
NOISE												
Noise Density	10 Hz – 400 Hz, 5 V		1.1	3		1.4	3	X		1.8	3.5	mg/√Hz
								Y		1.1	3	mg/√Hz
Clock Noise			5			5				5		mV p-p
FREQUENCY RESPONSE	2-pole Bessel											
−3 dB Frequency		360	400	440	360	400	440		360	400	440	Hz
−3 dB Frequency Drift	25°C to T_{MIN} or T_{MAX}		2			2				2		Hz
SELF-TEST												
Output Change (Cube vs. V_{DD})[3]	V_{DD} = 5 V	440	550	660	304	380	456	X	216	270	324	mV
								Y	440	550	660	mV
Logic Input High	V_{DD} = 5 V	3.5			3.5				3.5			V
Logic Input Low	V_{DD} = 5 V			1			1				1	V
Input Resistance	Pull-down resistor to GND	30	50		30	50			30	50		kΩ
OUTPUT AMPLIFIER												
Output Voltage Swing	$I_{OUT} = \pm 400\ \mu A$	0.25		V_{DD} − 0.25	0.25		V_{DD} − 0.25		0.25		V_{DD} − 0.25	V
Capacitive Load Drive		1000			1000				1000			pF
PREFILTER HEADROOM			280			400				560		*g*
CFSR @ 400 kHz			6			4.5				3		V/V
										6		V/V
POWER SUPPLY (V_{DD})		4.75		5.25	4.75		5.25		4.75		5.25	V
Functional Range		3.5		6	3.5		6		3.5		6	V
Quiescent Supply Current	V_{DD} = 5 V		2.2	2.9		2.2	2.9			2.2	2.9	mA
TEMPERATURE RANGE		−40		+105	−40		+105		−40		+105	°C

[1] All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.
[2] Zero *g* output is ratiometric.
[3] Self-test output at V_{DD} = (Self-Test Output at 5 V) × (V_{DD}/5 V)^3.

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Acceleration (Any Axis, Unpowered)	4,000 *g*
Acceleration (Any Axis, Powered)	4,000 *g*
V_S	−0.3 V to +7.0 V
All Other Pins	(COM − 0.3 V) to (V_S + 0.3 V)
Output Short-Circuit Duration (Any Pin to Common)	Indefinite
Operating Temperature Range	−65°C to +150°C
Storage Temperature	−65°C to +150°C

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although this product features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

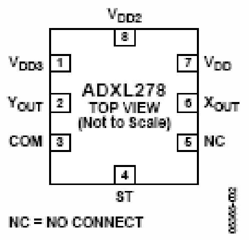


Figure 2. Pin Configuration

Table 3. Pin Function Descriptions

Pin No.	Mnemonic	Description
1	V_{DD3}	3.5 V to 6 V
2	Y_{OUT}	Y Channel Output
3	COM	Common
4	ST	Self-Test
5	NC	Do Not Connect
6	X_{OUT}	X Channel Output
7	V_{DD}	3.5 V to 6 V
8	V_{DD2}	3.5 V to 6 V

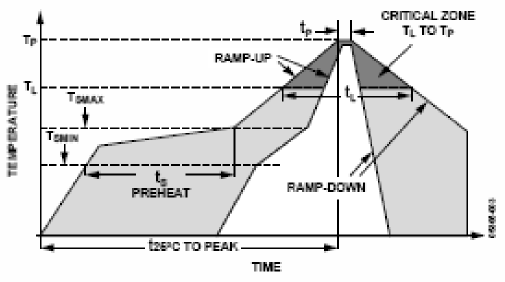


Figure 3. Recommended Soldering Profile

Table 4. Recommended Soldering Profile

Profile Feature	Sn63/Pb37	Pb-Free
AVERAGE RAMP RATE (T_L TO T_P)	3°C/s max	3°C/s max
PREHEAT		
Minimum Temperature (T_{SMIN})	100°C	150°C
Maximum Temperature (T_{SMAX})	150°C	200°C
TIME (T_{SMIN} TO T_{SMAX}), t_S	60 s – 120 s	60 s – 150 s
T_{SMAX} TO T_L		
Ramp-Up Rate	3°C/s	3°C/s
TIME MAINTAINED ABOVE LIQUIDOUS (T_L)		
Liquidous Temperature (T_L)	183°C	217°C
Time (t_L)	60 s – 150 s	60 s – 150 s
PEAK TEMPERATURE (T_P)	240°C + 0°C/–5°C	260°C + 0°C/–5°C
TIME WITHIN 5°C OF ACTUAL PEAK TEMPERATURE (t_P)	10 s – 30 s	20 s – 40 s
RAMP-DOWN RATE	6°C/s max	6°C/s max
TIME 25°C TO PEAK TEMPERATURE	6 min max	8 min max

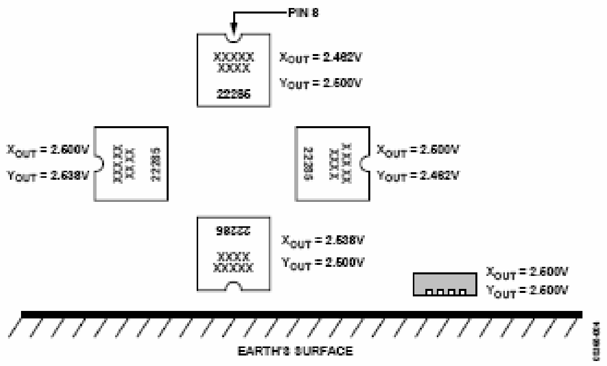


Figure 4. Output Response vs. Orientation

THEORY OF OPERATION

The ADXL278 provides a fully differential sensor structure and circuit path, resulting in the industry's highest resistance to EMI/RFI effects. This latest generation uses electrical feedback with zero-force feedback for improved accuracy and stability. The sensor resonant frequency is significantly higher than the signal bandwidth set by the on-chip filter, avoiding the signal analysis problems caused by resonant peaks near the signal bandwidth.

Figure 5 is a simplified view of one of the differential sensor elements. Each sensor includes several differential capacitor unit cells. Each cell is composed of fixed plates attached to the substrate and movable plates attached to the frame. Displacement of the frame changes the differential capacitance, which is measured by the on-chip circuitry.

Complementary 200 kHz square waves drive the fixed plates. Electrical feedback adjusts the amplitudes of the square waves such that the ac signal on the moving plates is 0. The feedback signal is linearly proportional to the applied acceleration. This unique feedback technique ensures that there is no net electrostatic force applied to the sensor. The differential feedback control signal is also applied to the input of the filter, where it is filtered and converted to a single-ended signal.

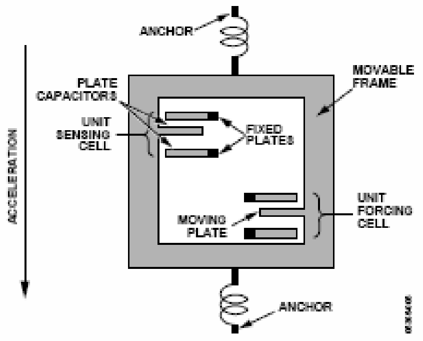


Figure 5. Simplified View of Sensor Under Acceleration

APPLICATIONS

POWER SUPPLY DECOUPLING

For most applications, a single 0.1 µF capacitor, C_{DC}, adequately decouples the accelerometer from noise on the power supply. However, in some cases, particularly where noise is present at the 200 kHz internal clock frequency (or any harmonic thereof), noise on the supply can cause interference on the ADXL278's output. If additional decoupling is needed, a 50 Ω (or smaller) resistor or ferrite bead cany be inserted in the supply line. Additionally, a larger bulk bypass capacitor (in the 1 µF to 4.7 µF range) can be added in parallel to C_{DC}.

SELF-TEST

The fixed fingers in the forcing cells are normally kept at the same potential as that of the movable frame. When the self-test digital input is activated, the voltage on the fixed fingers on one side of the moving plate in the forcing cells is changed. This creates an attractive electrostatic force, which causes the frame to move towards those fixed fingers. The entire signal channel is active; therefore, the sensor displacement causes a change in V_{OUT}. The ADXL278's self-test function is a comprehensive method of verifying the operation of the accelerometer.

Because electrostatic force is independent of the polarity of the voltage across capacitor plates, a positive voltage is applied in half of the forcing cells, and its complement in the other half of the forcing cells. Activating self-test causes a step function force to be applied to the sensor, while the capacitive coupling term is canceled. The ADXL278 has improved self-test functionality, including excellent transient response and high speed switching capabilities. Arbitrary force waveforms can be applied to the sensor by modulating the self-test input, such as test signals to measure the system frequency response or even crash signals to verify algorithms within the limits of the self-test swing.

The ST pin should never be exposed to voltages greater than $V_S + 0.3$ V. If this cannot be guaranteed due to the system design (for instance, if there are multiple supply voltages), then a low V_F clamping diode between ST and V_S is recommended.

CLOCK FREQUENCY SUPPLY RESPONSE

In any clocked system, power supply noise near the clock frequency may have consequences at other frequencies. An internal clock typically controls the sensor excitation and the signal demodulator for micromachined accelerometers.

If the power supply contains high frequency spikes, they may be demodulated and interpreted as an acceleration signal. A signal appears as the difference between the noise frequency and the demodulator frequency. If the power supply spikes are 100 Hz away from the demodulator clock, there is an output term at 100 Hz. If the power supply clock is at exactly the same frequency as the accelerometer clock, the term appears as an offset.

If the difference frequency is outside of the signal bandwidth, the filter attenuates it. However, both the power supply clock and the accelerometer clock may vary with time or temperature, which can cause the interference signal to appear in the output filter bandwidth.

The ADXL278 addresses this issue in two ways. First, the high clock frequency eases the task of choosing a power supply clock frequency such that the difference between it and the accelerometer clock remains well outside of the filter bandwidth. Second, the ADXL278 is the only micromachined accelerometer to have a fully differential signal path, including differential sensors. The differential sensors eliminate most of the power supply noise before it reaches the demodulator. Good high frequency supply bypassing, such as a ceramic capacitor close to the supply pins, also minimizes the amount of interference.

The clock frequency supply response (CFSR) is the ratio of the response at V_{OUT} to the noise on the power supply near the accelerometer clock frequency. A CFSR of 3 means that the signal at V_{OUT} is 3× the amplitude of an excitation signal at V_{DD} near the accelerometer internal clock frequency. This is analogous to the power supply response, except that the stimulus and the response are at different frequencies. The ADXL278's CFSR is 10× better than a typical single-ended accelerometer system.

SIGNAL DISTORTION

Signals from crashes and other events may contain high amplitude, high frequency components. These components contain very little useful information and are reduced by the 2-pole Bessel filter at the output of the accelerometer. However, if the signal saturates at any point, the accelerometer output does not look like a filtered version of the acceleration signal.

The signal may saturate anywhere before the filter. For example, if the resonant frequency of the sensor is low, the displacement per unit acceleration is high. The sensor may reach the mechanical limit of travel if the applied acceleration is high enough. This can be remedied by locating the accelerometer where it does not see high values of acceleration and by using a higher resonant frequency sensor, such as the ADXL278.

Also, the electronics may saturate in an overload condition between the sensor output and the filter input. Ensuring that internal circuit nodes operate linearly to at least several times the full-scale acceleration value can minimize electrical saturation. The ADXL278 circuit is linear to approximately 8× full scale.

OUTLINE DIMENSIONS

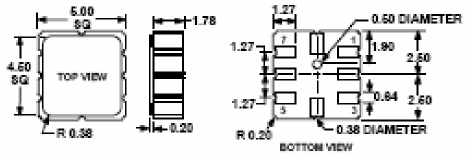


Figure 6. 8-Terminal Ceramic Leadless Chip Carrier [LCC]
(E-8)
Dimensions shown in millimeters

AXL278 ORDERING GUIDE

Model[1]	Parts per Reel	Measurement Range	Specified Voltage (V)	Temperature Range	Package Description	Package Option
AD22284-A-R2	250	±35 *g*/±35 *g*	5	–40°C to +105°C	8-Lead Ceramic Leadless Chip Carrier	E-8
AD22284-A	3000	±35 *g*/±35 *g*	5	–40°C to +105°C	8-Lead Ceramic Leadless Chip Carrier	E-8
AD22285-R2	250	±50 *g*/±50 *g*	5	–40°C to +105°C	8-Lead Ceramic Leadless Chip Carrier	E-8
AD22285	3000	±50 *g*/±50 *g*	5	–40°C to +105°C	8-Lead Ceramic Leadless Chip Carrier	E-8
AD22286-R2	250	±70 *g*/±35 *g*	5	–40°C to +105°C	8-Lead Ceramic Leadless Chip Carrier	E-8
AD22286	3000	±70 *g*/±35 *g*	5	–40°C to +105°C	8-Lead Ceramic Leadless Chip Carrier	E-8

[1] All models are on tape and reel and are Pb-free parts.